Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2016, Article ID 1479692, 13 pages
http://dx.doi.org/10.1155/2016/1479692

Research Article

Understanding Open Source Software Evolution Using Fuzzy Data Mining Algorithm for Time Series Data

Munish Saini,[1] Sandeep Mehmi,[2] and Kuljit Kaur Chahal[1]

[1]*Department of Computer Science, Guru Nanak Dev University, Amritsar, India*
[2]*Department of Computer Science, I.K.G. Punjab Technical University, Jalandhar, Punjab, India*

Correspondence should be addressed to Munish Saini; munish_1_saini@yahoo.co.in

Received 6 June 2016; Accepted 20 July 2016

Academic Editor: Gözde Ulutagay

Source code management systems (such as Concurrent Versions System (CVS), Subversion, and git) record changes to code repositories of open source software projects. This study explores a fuzzy data mining algorithm for time series data to generate the association rules for evaluating the existing trend and regularity in the evolution of open source software project. The idea to choose fuzzy data mining algorithm for time series data is due to the stochastic nature of the open source software development process. Commit activity of an open source project indicates the activeness of its development community. An active development community is a strong contributor to the success of an open source project. Therefore commit activity analysis along with the trend and regularity analysis for commit activity of open source software project acts as an important indicator to the project managers and analyst regarding the evolutionary prospects of the project in the future.

1. Introduction

Understanding software evolution in general and open source software (OSS) evolution in particular has been of wide interest in the recent past. A wide range of research studies have analysed OSS project evolution from different points of views such as growth [1], quality [2], and group dynamics [3]. However, there are a very few studies on commit activity in OSS projects. A commit is a change to a source code entity submitted by a developer through a source code management (SCM) system. SCM systems, such as Subversion (SVN) and git [4], manage the source code files of OSS systems and maintain log of each change (a.k.a. commit) made to the files. Committing is an important activity of the OSS development approach. Most of the OSS developers being volunteers, the success of these OSS projects is mainly determined by the committing activities of developers [5]. Commit activity indicates project activity which is further related to project success [6, 7]. Stakeholders of an OSS project, such as project managers, developers, and users, are interested in its future change behavior. Analysing the commit activity of an OSS project for finding trend and regularities in the evolution helps in indicating the future change behavior of the project and helps in decision making as far as project usage and management are concerned.

The OSS development is a stochastic process. Unlike the traditional development in which the environment is controlled, OSS development is based on contributions from volunteers who could not be forced to work even if something is of high priority for the project [1]. Along with this unplanned activity, there is a lack of planned documentation related to requirements and detailed design [8]. Classical time series techniques are inappropriate for analysis and forecasting of the data which involves random variables [8, 9]. Fuzzy time series can work for domains which involve uncertainty.

Commit activity of an OSS project is measured with the number of commits per month metric [5]. Kemerer and Slaughter [9] and Mockus and Votta [10] emphasize that the commits available in a SCM system (such as git [4]) can be used as a metric to study the evolution of OSS systems, and these studies motivate us too to choose number of commits per month as a metric to analyse and predict software evolution.

In this research work, the hybrid approach proposed by Chen et al. [11] (fuzzy theory along with data mining

algorithm) is used to generate linguistic rules from the time series data. In [11], Chen et al. specified the finding application for their algorithm and validating it as the future work. This present study considers both of these issues as one of the objectives. In this proposed work we divide the considered time series dataset into two data subsets, that is, training set and remaining set. We use the algorithm first to generate the association rules from the training set and then validate the accuracy and prediction capability of these rules on the remaining set. The high prediction accuracy indicates that the commits in the remaining set have regularity and trend in the number of commits performed for Eclipse CDT. The low prediction accuracy specifies that the commits on remaining set are not consistent with those of training set and there is irregularity and detrend in the commits performed.

Main objective of the present study is to explore the fuzzy data mining approach for time series used to generate the association rules for evaluating the existing trend and regularity in the evolution of OSS projects. As commit activity is a good indicator of continuous development activity of an OSS project, another objective of the present work is to develop a commit prediction model for OSS systems. Project managers, developers, and users can use the commit prediction model to understand the future commit activity of an OSS project and then plan schedule and allocate resources accordingly as per their role.

The rest of the paper is as follows. Section 2 presents the related work to study software evolution and the used fuzzy data mining technique. Section 3 explains the research methodology. Section 4 presents details of the experimental setup used for performing the study. Section 5 gives the results and analysis. Threats to validity of the results are explained in Section 6. Last section concludes the paper.

2. Related Work

The idea to analyse and predict the software evolution was seen initially in the late 1980s, when Yuen published his papers on the subject in a series of conferences on software maintenance [12–14]. He used time series analysis as a technique for software evolution prediction. Later, several studies used time series analysis for predicting software evolution. The software evolution metrics undertaken for prediction include the monthly number of changes [8, 9], change requests [15, 16], size and complexity [17, 18], defects [19, 20], clones [21], and maintenance effort [22].

Kemerer and Slaughter [9] looked at the evolution of two proprietary systems using two approaches: one based on the time series analysis (ARIMA) and the other based on a technique called sequence analysis. They found ARIMA models inappropriate for analysis as the dataset was largely random in nature. Antoniol et al. [21] presented an approach for monitoring and predicting evolution of software clones across subsequent versions of a software system (*mSQL*) using time series analysis (ARIMA).

Caprio et al. [17] used time series analysis to estimate size and complexity of the Linux Kernel. They used ARIMA model to predict evolution of the Linux Kernel by using dataset related to 68 stable releases of the software system.

Herraiz et al. [8] applied a stationery model based on time series analysis to the monthly number of changes in the CVS repository of Eclipse. Their model predicted the number of changes per month for the next three months. They employed Kernel smoothing to reduce noise, a lesson learned from Kemerer and Slaughter [9] study who could not get good results of ARIMA modelling for predicting number of changes, as they ignored noise present in the data.

Kenmei et al. [16] applied ARIMA to model and forecast change requests per unit of size of large open source projects. Data from three large open source projects, Mozilla, JBoss, and Eclipse, confirm the capability of the approach to effectively perform prediction and identify trends. They report the evidence that ARIMA models almost always outperform the predictive accuracy of simple models such as linear regression or random walk [16]. The benchmark models selected by Kenmei et al. [16] for evaluating the prediction accuracy of ARIMA model are not rigorous.

Raja et al. [20] did time series analysis of defect reports of eight open source software projects over a period of five years and found ARIMA (0, 1, 1) model to be useful for defect prediction. Kläs et al. [19] combined time series analysis with expert opinion to create prediction models for defects. They suggest that a hybrid model is more powerful than data models in the early phases of a project's life cycle.

Goulão et al. [15] used time series technique for long-term prediction of the overall number of change requests. They investigated the suitability of ARIMA model to predict the long-term fluctuation of all change requests for a project having seasonal patterns, such as Eclipse. They found that their ARIMA model is statistically more significant and outperforms the nonseasonal models.

Amin et al. [23] used the ARIMA model in place of software reliability growth model (SRGM) to predict the software reliability. SRGM has restrictive assumption on environment of the software under analysis. They specified that the ARIMA modelling is far better than SRGM approach as ARIMA is data oriented and cover all limitations of the previous approaches.

A perusal of the existing research in this area shows ARIMA modelling as the most frequently used prediction procedure. However, OSS development is a stochastic process. Unlike the traditional development in which the environment is controlled, OSS development is based on contributions from volunteers who could not be forced to work even if something is of high priority for the project [1]. Along with this unplanned activity, there is a lack of planned documentation related to requirements and detailed design [8].

Open source projects, without any tight organizational support, face many uncertainties. Uncertainty lies in an uncontrolled development environment such as availability of contributors at any point of time. Due to uncertainty, there is a large fluctuation in consecutive values (as observed in monthly commit data of Eclipse CDT in Figure 1). Most of the classical time series techniques are inappropriate for analysis and forecasting of the data which involve uncertainty [8, 9]. Research literature also indicates that ARIMA modelling can

Table 1: Descriptive statistics of the open source software projects.

Software	Origin date	Data collection date	Number of years	Number of authors	Total commits analysed
Eclipse CDT	6/27/2002	9/23/2013	11	135	26246

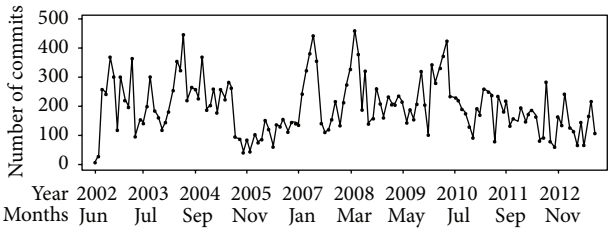

Figure 1: Eclipse CDT commit activity.

be useful when there is uncertainty in the data, but only after applying smoothing to reduce noise [24].

Hong et al. [25] introduced the fuzzy data mining algorithm for quantitative values. The algorithm extracts the useful knowledge from the transactional database having quantitative values. The algorithm combines the fuzzy set concept with that of Apriori algorithm.

Chen et al. [11] extended the work of Hong et al. and analysed the fuzzy data mining algorithm on time series data. They proposed an approach in which the concepts of fuzzy sets are used along with the data mining Apriori algorithm to generate linguistic association rules. They use the fuzzy membership function to convert the time series data to fuzzy set and then apply the Apriori algorithm to generate association rules accordingly. They specified the validation and finding of applications for their algorithm as the future work. So, with the continuation of their work, we are using algorithm for analysing the commit activity of open source software project for finding existing regularity and trend in the commit activity of OSS project.

Suresh and Raimond [26] extended the work of Chen et al. [11]. They proposed a new algorithm called extended fuzzy frequent pattern algorithm for analysing the time series data. The association rules are generated without generating the candidate sets.

This paper uses a fuzzy data mining approach on time series [11] to analyse the commit activity in open source software projects. The research questions that this study aims to answer are (1) analysing the commit activity of OSS project for finding the trend and regularity and (2) validating the fuzzy data mining algorithm on OSS project data.

3. Methodology

The objective of this empirical study is to investigate the suitability of the fuzzy data mining method to analyse the number of commits per month as a software system evolves for finding the regularity and trend. This section describes the data collection process, basic concepts of fuzzy time series, and the fuzzy data mining method.

3.1. Data Collection. The development repository of open source software project (Eclipse CDT) is obtained from GIT Hub [27]. A repository is downloaded by making the clone of the original repository onto the local machine by using GIT Bash [4]. A script is written in JAVA to fetch the number of commits per month for the observation period for all the software projects. The descriptive statistics about the development repositories of all software projects is shown in Table 1.

Eclipse [28] is an integrated development environment. Eclipse has base workspace and extendable plug-in for customizing the development environment. It is used to develop application in different languages such as JAVA, C/C++, COBOL, and PHP, by using available plug-in. The two variations Eclipse SDK and Eclipse CDT are well known for developing applications. Eclipse SDK is compatible with JAVA and used by JAVA developers for building project on Eclipse platform. Eclipse CDT provides C/C++ development tooling [29]. Eclipse CDT allows developing application in C/C++ using Eclipse. Eclipse CDT provides various features [30] such as full featured editor, debugging, refactoring, parser, and indexes. In this study we consider Eclipse CDT only. The number of commits for each month from 6/27/2002 to 9/23/2013 is arranged month-wise to form a time series (for 136 months) shown in Figure 1.

3.2. Basics of Fuzzy Set Theory and Fuzzy Time Series. The concept of fuzzy set was introduced by Zadeh [31] in 1965. It was the extension of classical set theory. The fuzzy set is characterized by degree of membership function [31]. The membership function can be of various forms such as triangular, L-function, R-function, and trapezoidal function used depending upon the application and requirement. The present study uses both L and R membership function to define the values of fuzzy variable.

3.3. Apriori Algorithm. Apriori algorithm [32, 33] is one of the classical algorithms proposed by R. Srikant and R. Agrawal in 1994 for finding frequent patterns for generating association rules. Apriori employs an iterative approach known as level-wise search, where k-itemsets are used to explore $(k + 1)$-itemsets.

Apriori algorithm is executed in two steps. Firstly it retrieves all the frequent itemsets whose support is not smaller than the minimum support (min_sup). The first step further consists of join and pruning action. In joining, the candidate set C_k is produced by joining L_{k-1} with itself. In pruning, the candidate sets are pruned by applying the Apriori property; that is, all the nonempty subsets of frequent itemset must also be frequent.

The pseudocode for generation of frequent itemsets is as follows:

C_k: Candidate itemset of size k

L_k: Frequent itemset of size k

L_1 = frequent 1-itemset

For ($k = 1$; $L_k! = \Phi$; $k++$)
{
C_{k+1} = Join L_k with L_k to generate C_{k+1};
L_{k+1} = Candidate in C_{k+1} with support greater than or equal to min support;
}
End;
Return L_k;

Next, it uses the frequent itemsets to generate the strong association rules satisfying the minimum confidence (min_conf) threshold. The pseudocode for generation of strong association rules is as follows:

Input:
Frequent Itemset, L
Minimum confidence threshold, min_conf
Output: Strong association rules, R
$R = \Phi$
for each frequent itemset I in L
{
for each non-empty subset s of I
{
conf ($S \rightarrow I - S$) = support_count (I)/support_count (S)
if conf >= min_conf
{
// generate strong association rule
Rule r = "$s \rightarrow (I - S)$"
$R = RU\{r\}$
}
}
}

3.4. Fuzzy Data Mining Algorithm for Time Series Data. Chen et al. [11] extended the work of Hong et al. [25] and proposed the fuzzy data mining for time series data. The time series data of K points are entered as input along with the predefined minimum support λ, minimum confidence α, and window size of w.

The input data is first converted to generate ($K - w + 1$) sequences; each subsequence has w elements. The fuzzy membership function is used to convert each data item into the equivalent fuzzy set. The Apriori algorithm is used to mine frequent fuzzy sets. Moreover, the data reduction method is used to remove the redundant data items.

The association rules are generated in the same way as generated in Apriori algorithm. The stepwise process of the fuzzy Apriori algorithm is given below.

Input:
Time series with K data points
Membership function values h
Minimum support λ
Minimum confidence α
Sliding window size w
Fuzzy set f_p
Output: Set of fuzzy association rules

Step 1. Convert the time series data into ($K-w+1$) sequences, where each sequence has maximum of w elements. Suppose we assume $w = 5$; then each sequence has maximum of 5 elements. The elements of subsequence are referred to as data variable and given as A_1, A_2, A_3, A_4, and A_5, respectively.

Step 2. Apply fuzzy membership function on elements of time series to generate fuzzy set (f_p).

Step 3. Based on the membership function and its user defined level (suppose low, middle, and high), each data variable after conversion to fuzzy item lies in different user defined levels (such as $A_{1.\text{Low}}$, $A_{2.\text{Middle}}$, and $A_{3.\text{High}}$ referred to as fuzzy items).

Step 3. Calculate the scalar cardinality count of each fuzzy item of subsequences.

Step 4. Compare the total scalar cardinality count of each subsequence with the minimum support value. The sequences with value greater than or equal to α are kept in L_1.

The support value of subsequence is generated as

$$\text{Support value} = \frac{\text{Count}}{K - w + 1}. \quad (1)$$

Step 5. If L_1 = NULL, then exit; else do the following step for $r = 1$ to K.

Step 6. Join L_r with L_r to generate candidate ($r + 1$) fuzzy itemset (i.e., C_{r+1}) (similar to Apriori algorithm except not joining the items generated from same order of data point and join is possible only if ($r - 1$) data items in both sets are the same).

(i) Calculate the fuzzy value of each candidate fuzzy itemset by using fuzzy set theory, that is, Min ($f_1 \Lambda f_2 \Lambda \ldots f_k$).

(ii) Count the scalar cardinality of each fuzzy candidate itemset.

(iii) If count is more than or equal to α, then put it in L_{r+1} and calculate its support value as

$$\text{Support value} = \frac{\text{Count}}{K - w + 1}. \quad (2)$$

Step 7. If L_{r+1} = NULL, then exit; otherwise go to Step 6 again.

Step 8. Remove redundant large itemset (i.e., by shifting each large itemsets ($I_1, I_2, I_3, \ldots, I_q$) into ($I_1', I_2', I_3', \ldots, I_q'$) such that fuzzy region R_{11} becomes R_{11}' when shifted).

Table 2: Fuzzy candidate item sets in C_2.

$A_{1.\text{Low}} \cap A_{2.\text{Low}}$	$A_{1.\text{Low}} \cap A_{2.\text{Middle}}$	$A_{1.\text{Low}} \cap A_{3.\text{Low}}$	$A_{1.\text{Low}} \Lambda A_{3.\text{Middle}}$	$A_{1.\text{Low}} \Lambda A_{4.\text{Low}}$
$A_{1.\text{Low}} \cap A_{4.\text{Middle}}$	$A_{1.\text{Low}} \cap A_{5.\text{Low}}$	$A_{1.\text{Low}} \cap A_{5.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{2.\text{Low}}$	$A_{1.\text{Middle}} \cap A_{2.\text{Middle}}$
$A_{1.\text{Middle}} \cap A_{3.\text{Low}}$	$A_{1.\text{Middle}} \cap A_{3.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{4.\text{Low}}$	$A_{1.\text{Middle}} \cap A_{4.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{5.\text{Low}}$
$A_{1.\text{Middle}} \cap A_{5.\text{Middle}}$	$A_{2.\text{Low}} \cap A_{3.\text{Low}}$	$A_{2.\text{Low}} \cap A_{3.\text{Middle}}$	$A_{2.\text{Low}} \cap A_{4.\text{Low}}$	$A_{2.\text{Low}} \cap A_{4.\text{Middle}}$
$A_{2.\text{Low}} \cap A_{5.\text{Low}}$	$A_{2.\text{Low}} \cap A_{5.\text{Middle}}$	$A_{2.\text{Middle}} \cap A_{3.\text{Low}}$	$A_{2.\text{Middle}} \cap A_{3.\text{Middle}}$	$A_{2.\text{Middle}} \cap A_{4.\text{Low}}$
$A_{2.\text{Middle}} \cap A_{4.\text{Middle}}$	$A_{2.\text{Middle}} \cap A_{5.\text{Low}}$	$A_{2.\text{Middle}} \cap A_{5.\text{Middle}}$	$A_{3.\text{Low}} \cap A_{4.\text{Low}}$	$A_{3.\text{Low}} \cap A_{4.\text{Middle}}$
$A_{3.\text{Low}} \cap A_{5.\text{Low}}$	$A_{3.\text{Low}} \cap A_{5.\text{Middle}}$	$A_{3.\text{Middle}} \cap A_{4.\text{Low}}$	$A_{3.\text{Middle}} \cap A_{4.\text{Middle}}$	$A_{3.\text{Middle}} \cap A_{5.\text{Low}}$
$A_{3.\text{Middle}} \cap A_{5.\text{Middle}}$	$A_{4.\text{Low}} \cap A_{5.\text{Low}}$	$A_{4.\text{Low}} \cap A_{5.\text{Middle}}$	$A_{4.\text{Middle}} \cap A_{5.\text{Low}}$	$A_{4.\text{Middle}} \cap A_{5.\text{Middle}}$

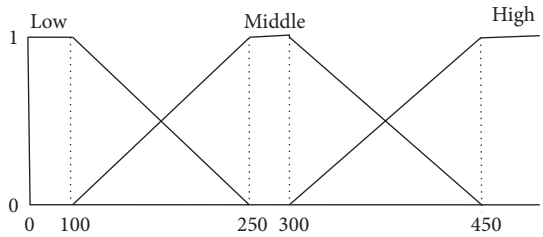

Figure 2: Membership function.

Step 9. Generate the association rules using Apriori rule generation method and calculate the confidence of each rule. The only variation is that in place of normal data itemset we have fuzzy itemset, so concept of intersection is used in rule not union. If confidence value is not less than minimum confidence α, then keep the rule; otherwise reject.

4. Experimental Setup

The experiment is performed on x86 Family 6 Model 15 Stepping 6 GenuineIntel ~2131 Mhz Processor 1 GB RAM, Microsoft Windows XP Professional operating system, version 5.1.2600 Service Pack 3 Build 2600, 240 GB hard disk. The fuzzy data mining algorithm for time series data is implemented using JAVA.

5. Results and Analysis

The dataset is divided into two subsets: training dataset (120 months) and remaining dataset (16 months). The complete process of generating the association rule using fuzzy data mining algorithm is performed in two steps.

(A) The association rules are generated on training dataset using fuzzy data mining for time series data algorithm.

(B) The validation of generated association rule is done using training and remaining dataset.

5.1. Generation of Association Rules from Training Dataset. The fuzzy data mining algorithm of time series [11] is applied on training dataset to generate association rule. We assumed $w = 5$, $\lambda = 25\%$ (25/100 $*$ number of subsequences), and $\alpha = 65\%$; membership function and its values are shown in Figure 2. These assumptions are made by referring to and understanding the concepts of fuzzy data mining algorithm given in [11]. The commits are divided according to the level of activity. The commits in the range of 0–100, 250–300, and 450 onwards indicate the *low*, *middle* (average), and *high* level of activity, respectively. The level of commit activity indicates the amount of work done in a commit.

(i) Generation of Subsequences

We have $K = 120$ (as number of months)

$w = 5$ (window size)

The number of subsequences is generated as $(K-w+1)$

Therefore number of subsequences $= (K - w + 1) = (120 - 5 + 1) = 116$

All generated subsequences for $K = 120$ (shown in Table 8 of the Appendix).

(ii) Transformation of Data to Fuzzy Sets. In this step, we transform the commit activity data into fuzzy set (using membership function) and count the scalar cardinality of each data variable (shown in Table 9 of the Appendix).

All these data variables are considered as candidate itemsets (C_1).

For generation of L_1, data variables having count more than $\lambda = 25\%$ (25/100 $*$ 116) $= 29$ are considered.

It is found that L_1 contain $A_{1.\text{Low}}$, $A_{1.\text{Middle}}$, $A_{2.\text{Low}}$, $A_{2.\text{Middle}}$, $A_{3.\text{Low}}$, $A_{3.\text{Middle}}$, $A_{4.\text{Low}}$, $A_{4.\text{Middle}}$, $A_{5.\text{Low}}$, and $A_{5.\text{Middle}}$.

Further, calculate support value of each candidate itemset using

$$\text{Support value} = \frac{\text{Count}}{K - w + 1}. \tag{3}$$

(iii) Generation of C_2 and L_2. For generating C_2 (shown in Table 2), join L_1 with L_1, not joining the items generated from same order of data points; that is, $A_{1.\text{Low}}$ join with $A_{1.\text{Middle}}$ is not allowed, similar to all others also. Meanwhile, joining all the properties of Apriori algorithm is used.

Next, use Min $(f_1 \cap f_2)$ function to find the value of each of the candidate fuzzy sets. Count the scalar cardinality of each of the candidate sets in C_2.

Table 3: Fuzzy item sets in L_2.

$A_{1.\text{Low}} \cap A_{2.\text{Low}}$	$A_{1.\text{Low}} \cap A_{3.\text{Low}}$	$A_{1.\text{Low}} \cap A_{4.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{2.\text{Middle}}$	$A_{1.\text{Middle}} \cap_{3.\text{Middle}}$
$A_{1.\text{Middle}} \cap_{4.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{5.\text{Low}}$	$A_{1.\text{Middle}} \cap A_{5.\text{Middle}}$	$A_{2.\text{Low}} \cap A_{3.\text{Low}}$	$A_{2.\text{Low}} \cap A_{4.\text{Low}}$
$A_{2.\text{Middle}} \cap_{3.\text{Middle}}$	$A_{2.\text{Middle}} \cap A_{4.\text{Middle}}$	$A_{2.\text{Middle}} \cap A_{5.\text{Middle}}$	$A_{3.\text{Low}} \cap A_{4.\text{Low}}$	$A_{3.\text{Low}} \cap A_{5.\text{Low}}$
$A_{3.\text{Middle}} \Lambda A_{4.\text{Middle}}$	$A_{3.\text{Middle}} \cap A_{5.\text{Middle}}$	$A_{4.\text{Low}} \cap A_{5.\text{Low}}$	$A_{4.\text{Middle}} \cap A_{5.\text{Middle}}$	

Table 4: Fuzzy item sets in L_6.

$A_{1.\text{Middle}} \cap A_{2.\text{Middle}} \cap A_{3.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{2.\text{Middle}} \cap A_{4.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{2.\text{Middle}} \cap A_{5.\text{Middle}}$
$A_{1.\text{Middle}} \cap A_{3.\text{Middle}} \cap A_{4.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{3.\text{Middle}} \cap A_{5.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{4.\text{Middle}} \cap A_{5.\text{Middle}}$
$A_{2.\text{Middle}} \cap A_{3.\text{Middle}} \cap A_{4.\text{Middle}}$	$A_{2.\text{Middle}} \cap A_{3.\text{Middle}} \cap A_{5.\text{Middle}}$	$A_{2.\text{Middle}} \cap A_{4.\text{Middle}} \cap A_{5.\text{Middle}}$
$A_{3.\text{Middle}} \cap A_{4.\text{Middle}} \cap A_{5.\text{Middle}}$		

Table 5: Largest fuzzy item sets generated.

$A_{1.\text{Middle}} \cap A_{2.\text{Middle}} \cap A_{3.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{2.\text{Middle}} \cap A_{4.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{2.\text{Middle}} \cap A_{5.\text{Middle}}$
$A_{1.\text{Middle}} \cap A_{3.\text{Middle}} \cap A_{4.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{3.\text{Middle}} \cap A_{5.\text{Middle}}$	$A_{1.\text{Middle}} \cap A_{4.\text{Middle}} \cap A_{5.\text{Middle}}$

The candidate fuzzy sets with a value not less than the threshold value are kept in L_2 and also find their support value. Now, L_2 contain fuzzy itemsets shown in Table 3.

(iv) Generation of C_3 and L_3. For generating C_3, join L_2 with L_2, not joining the items generated from same order of data points. In case of C_3, join is possible between only those fuzzy sets where at least one data item is common in both. After generation of C_3, only those fuzzy itemsets are put in L_3 (shown in Table 4) whose count is not less than threshold value.

(v) Generation of C_4. Join L_3 with L_3, not joining the items generated from same order of data point. Join is possible only for those fuzzy sets where at least two data items are common in both.

After generation of C_4, it is found that no element has count more than threshold value; therefore L_4 = Null.

(vi) Removal of Redundant Large Itemsets. Remove the redundant large itemsets from L_3 using Step 8 of the algorithm described in Section 3.4.

After applying Step 8 of algorithm, we are left with these large itemsets (shown in Table 5).

(vii) Generate Association Rules. Generate the association rules from these large itemsets using Step 8 of the algorithm described in Section 3.4. All the generated rules are shown in Table 6 where strong rules having count more than threshold confidence are marked as bold.

In this experiment we use $\alpha = 65\%$; it means only those rules are valid (and are called strong association rules) and have support value not less than 65%. We found 18 rules in this case; each rule acts as the knowledge base for the project manager and developers of the project. For example, the generated rule $\mathbf{A_{1.Middle}} \cap \mathbf{A_{2.Middle}} \rightarrow \mathbf{A_{3.Middle}}$ specifies that if the value of first and second data point lies in the middle, then there is high probability that the third data point has middle value also. All these rules act as a precise and compact knowledge for project manager and analyst.

5.2. Validation of Association Rules Using Training and Remaining Dataset

(i) Validation on Training Dataset. It is found that 65% of the transactions in the Eclipse CDT training set follow these rules. These rules act as a compact and concrete knowledge of this data.

(ii) Validation on Remaining Dataset. All generated association rules are tested on remaining dataset values to find the regularity and trend in the number of commits analysed to find whether the commits are performed at same rate or not. If the rate is same, then it means Eclipse has consistent growth; otherwise there exists a variation in the considered software growth. It is found that in case of remaining set the applicability of these rules decreases. This thing is again verified by generating the association rules from the remaining set. The generated rules from remaining dataset specify that the data in the remaining dataset is more towards lower range of commits performed.

Following are the factors due to which this variation in the commit rate is found:

(a) Most of the values in the remaining dataset are towards *low* range. This point is verified by finding the largest frequent fuzzy itemsets and then generating the association rules from the remaining dataset by using fuzzy data mining for time series data algorithm. The largest frequent itemset found is shown in Table 7.

The generated frequent itemsets consist of only data points with low range. Hence most of the entries of commits in remaining datasets are probably *low*. This specifies that the numbers of commits analysed from 6/1/2012 to 9/23/2013 is less as compared to the number of commits analysed in the training dataset. It specifies the less activity in the development of considered software growth in this period.

Table 6: Calculated confidence value of various association rules.

$\mathbf{A_{1.Middle} \cap A_{2.Middle} \rightarrow A_{3.Middle}}$	**0.741**	**74%**
$A_{3.Middle} \rightarrow A_{1.Middle} \cap A_{2.Middle}$	0.522	52%
$\mathbf{A_{1.Middle} \cap A_{3.Middle} \rightarrow A_{2.Middle}}$	**0.781**	**78%**
$A_{2.Middle} \rightarrow A_{1.Middle} \cap A_{3.Middle}$	0.526	53%
$\mathbf{A_{2.Middle} \cap A_{3.Middle} \rightarrow A_{1.Middle}}$	**0.735**	**74%**
$A_{1.Middle} \rightarrow A_{2.Middle} \cap A_{3.Middle}$	0.529	53%
$\mathbf{A_{1.Middle} \cap A_{2.Middle} \rightarrow A_{4.Middle}}$	**0.692**	**69%**
$A_{4.Middle} \rightarrow A_{1.Middle} \cap A_{2.Middle}$	0.491	49%
$\mathbf{A_{1.Middle} \cap A_{4.Middle} \rightarrow A_{2.Middle}}$	**0.762**	**76%**
$A_{2.Middle} \rightarrow A_{1.Middle} \cap A_{4.Middle}$	0.491	49%
$\mathbf{A_{2.Middle} \cap A_{4.Middle} \rightarrow A_{1.Middle}}$	**0.724**	**72%**
$A_{1.Middle} \rightarrow A_{2.Middle} \cap A_{4.Middle}$	0.494	49%
$\mathbf{A_{1.Middle} \cap A_{2.Middle} \rightarrow A_{5.Middle}}$	**0.695**	**69%**
$A_{5.Middle} \rightarrow A_{1.Middle} \cap A_{2.Middle}$	0.497	50%
$\mathbf{A_{1.Middle} \cap A_{5.Middle} \rightarrow A_{2.Middle}}$	**0.737**	**74%**
$A_{2.Middle} \rightarrow A_{1.Middle} \cap A_{5.Middle}$	0.493	49%
$\mathbf{A_{2.Middle} \cap A_{5.Middle} \rightarrow A_{1.Middle}}$	**0.759**	**76%**
$A_{1.Middle} \rightarrow A_{2.Middle} \cap A_{5.Middle}$	0.496	50%
$\mathbf{A_{1.Middle} \cap A_{3.Middle} \rightarrow A_{4.Middle}}$	**0.742**	**74%**
$A_{4.Middle} \rightarrow A_{1.Middle} \cap A_{3.Middle}$	0.499	50%
$\mathbf{A_{1.Middle} \cap A_{4.Middle} \rightarrow A_{3.Middle}}$	**0.775**	**78%**
$A_{3.Middle} \rightarrow A_{1.Middle} \cap A_{4.Middle}$	0.496	50%
$\mathbf{A_{3.Middle} \cap A_{4.Middle} \rightarrow A_{1.Middle}}$	**0.691**	**69%**
$A_{1.Middle} \rightarrow A_{3.Middle} \cap A_{4.Middle}$	0.502	50%
$\mathbf{A_{1.Middle} \cap A_{3.Middle} \rightarrow A_{5.Middle}}$	**0.751**	**75%**
$A_{5.Middle} \rightarrow A_{1.Middle} \cap A_{3.Middle}$	0.510	51%
$\mathbf{A_{1.Middle} \cap A_{5.Middle} \rightarrow A_{3.Middle}}$	**0.757**	**76%**
$A_{3.Middle} \rightarrow A_{1.Middle} \cap A_{5.Middle}$	0.502	50%
$\mathbf{A_{3.Middle} \cap A_{5.Middle} \rightarrow A_{1.Middle}}$	**0.738**	**74%**
$A_{1.Middle} \rightarrow A_{3.Middle} \cap A_{5.Middle}$	0.509	51%
$\mathbf{A_{1.Middle} \cap A_{4.Middle} \rightarrow A_{5.Middle}}$	**0.792**	**79%**
$A_{5.Middle} \rightarrow A_{1.Middle} \cap A_{4.Middle}$	0.515	51%
$\mathbf{A_{1.Middle} \cap A_{5.Middle} \rightarrow A_{4.Middle}}$	**0.764**	**76%**
$A_{4.Middle} \rightarrow A_{1.Middle} \cap A_{5.Middle}$	0.511	51%
$\mathbf{A_{4.Middle} \cap A_{5.Middle} \rightarrow A_{1.Middle}}$	**0.715**	**71%**
$A_{1.Middle} \rightarrow A_{4.Middle} \cap A_{5.Middle}$	0.513	51%

Table 7: Largest frequent fuzzy item set for remaining dataset.

$A_{1.Low} \cap A_{2.Low} \cap A_{3.Low} \cap A_{4.Low}$
$A_{1.Low} \cap A_{2.Low} \cap A_{3.Low} \cap A_{5.Low}$
$A_{1.Low} \cap A_{2.Low} \cap A_{4.Low} \cap A_{5.Low}$
$A_{1.Low} \cap A_{3.Low} \cap A_{4.Low} \cap A_{5.Low}$

(b) Factors like the number of active users and number of files changed (addition, deletion, and modification) are less.

6. Discussion

The fuzzy data mining algorithm for time series data [11] allows efficient mining of the association rules from the large

Table 8: Generated subsequences.

Sp	Subsequences				
1	9	25	252	240	361
2	25	252	240	361	298
3	252	240	361	298	118
4	240	361	298	118	294
5	361	298	118	294	219
6	298	118	294	219	193
7	118	294	219	193	359
8	294	219	193	359	96
9	219	193	359	96	151
10	193	359	96	151	141
11	359	96	151	141	198
12	96	151	141	198	296
13	151	141	198	296	183
14	141	198	296	183	160
15	198	296	183	160	117
16	296	183	160	117	139
17	183	160	117	139	179
18	160	117	139	179	255
19	117	139	179	255	351
20	139	179	255	351	320
21	179	255	351	320	440
22	255	351	320	440	219
23	351	320	440	219	260
24	320	440	219	260	254
25	440	219	260	254	227
26	219	260	254	227	363
27	260	254	227	363	185
28	254	227	363	185	200
29	227	363	185	200	254
30	363	185	200	254	175
31	185	200	254	175	253
32	200	254	175	253	221
33	254	175	253	221	281
34	175	253	221	281	263
35	253	221	281	263	87
36	221	281	263	87	83
37	281	263	87	83	42
38	263	87	83	42	80
39	87	83	42	80	44
40	83	42	80	44	99
41	42	80	44	99	71
42	80	44	99	71	83
43	44	99	71	83	150
44	99	71	83	150	120
45	71	83	150	120	60
46	83	150	120	60	132
47	150	120	60	132	127
48	120	60	132	127	151
49	60	132	127	151	109
50	132	127	151	109	141

Table 8: Continued.

Sp	Subsequences				
51	127	151	109	141	141
52	151	109	141	141	135
53	109	141	141	135	242
54	141	141	135	242	320
55	141	135	242	320	380
56	135	242	320	380	438
57	242	320	380	438	355
58	320	380	438	355	141
59	380	438	355	141	107
60	438	355	141	107	114
61	355	141	107	114	153
62	141	107	114	153	212
63	107	114	153	212	133
64	114	153	212	133	212
65	153	212	133	212	273
66	212	133	212	273	326
67	133	212	273	326	455
68	212	273	326	455	378
69	273	326	455	378	186
70	326	455	378	186	317
71	455	378	186	317	139
72	378	186	317	139	155
73	186	317	139	155	257
74	317	139	155	257	208
75	139	155	257	208	162
76	155	257	208	162	229
77	257	208	162	229	205
78	208	162	229	205	203
79	162	229	205	203	234
80	229	205	203	234	214
81	205	203	234	214	144
82	203	234	214	144	185
83	234	214	144	185	153
84	214	144	185	153	206
85	144	185	153	206	315
86	185	153	206	315	204
87	153	206	315	204	103
88	206	315	204	103	336
89	315	204	103	336	278
90	204	103	336	278	329
91	103	336	278	329	370
92	336	278	329	370	419
93	278	329	370	419	230
94	329	370	419	230	224
95	370	419	230	224	217
96	419	230	224	217	188
97	230	224	217	188	176
98	224	217	188	176	129
99	217	188	176	129	91
100	188	176	129	91	188

Table 8: Continued.

Sp	Subsequences				
101	176	129	91	188	169
102	129	91	188	169	256
103	91	188	169	256	245
104	188	169	256	245	237
105	169	256	245	237	74
106	256	245	237	74	228
107	245	237	74	228	179
108	237	74	228	179	212
109	74	228	179	212	128
110	228	179	212	128	154
111	179	212	128	154	150
112	212	128	154	150	192
113	128	154	150	192	148
114	154	150	192	148	171
115	150	192	148	171	181
116	192	148	171	181	163

dataset. These generated rules help in finding the regularity and existing trend for OSS projects. We have used the commit activity data of Eclipse CDT. The commits in repository are directly related to the activities such as file or code changed, deleted, or modified. By analysing the trend in commits data, we can interpret the development or evolution activity of the considered software. In the above experiment, the original dataset is divided into two subdatasets (training and remaining dataset).

The generated association rules from training set allow analysing the regularity and trends in the commits of OSS project. These rules also help in predicting and analysing the future evolution or development activity of the Eclipse CDT. The generated rules are validated on the remaining dataset to find its applicability. It is found that applicability of these rules on remaining set decreases. The results of the algorithm indicate that the commit activity of remaining dataset has *low* activity range. There may be other factors also which are the cause of this decrease behavior in the number of commits. These may include factors like the number of active users and number of files changed (addition, deletion, and modification) which are less.

7. Threats to Validity

This section discusses the threats to validity of the study.

Construct validity threats concern the relationship between theory and observation. These threats can be mainly due to the fact that we assumed all the commits posted in the revision control tool *git* [4]. Any changes performed in the source code, but not logged through the tool, may not have become part of the study.

Internal validity concerns the selection of subject systems and the analysis methods. This study uses a *month* as the unit of measure for tracking the types of change activities. In the future, we would like to use more natural and insightful partition based on major/minor versions of the OSS project

Table 9: Transformation of data elements into fuzzy set.

Count	47.595	60.227	8.178	47.275	60.547	8.178	46.802	61.02	8.178	47.262	60.56	8.178	47.775	60.047	8.178
Sp	$A_{1.Low}$	$A_{1.Middle}$	$A_{1.High}$	$A_{2.Low}$	$A_{2.Middle}$	$A_{2.High}$	$A_{3.Low}$	$A_{3.Middle}$	$A_{3.High}$	$A_{4.Low}$	$A_{4.Middle}$	$A_{4.High}$	$A_{5.Low}$	$A_{5.Middle}$	$A_{5.High}$
1	1	0	0	1	0	0	0	1	0	0.067	0.933	0	0	0.593	0.407
2	1	0	0	0	1	0	0.067	0.933	0	0	0.593	0.407	0	1	0
3	0	1	0	0.067	0.933	0	0	0.593	0.407	0	1	0	0.88	0.12	0
4	0.067	0.933	0	0	0.593	0.407	0	1	0	0.88	0.12	0	0	1	0
5	0	0.593	0.407	0	1	0	0.88	0.12	0	0	1	0	0.207	0.793	0
6	0	1	0	0.88	0.12	0	0	1	0	0.207	0.793	0	0.38	0.62	0
7	0.88	0.12	0	0	1	0	0.207	0.793	0	0.38	0.62	0	0	0.607	0.393
8	0	1	0	0.207	0.793	0	0.38	0.62	0	0	0.607	0.393	1	0	0
9	0.207	0.793	0	0.38	0.62	0	0	0.607	0.393	1	0	0	0.66	0.34	0
10	0.38	0.62	0	0	0.607	0.393	1	0	0	0.66	0.34	0	0.727	0.273	0
11	0	0.607	0.393	1	0	0	0.66	0.34	0	0.727	0.273	0	0.347	0.653	0
12	1	0	0	0.66	0.34	0	0.727	0.273	0	0.347	0.653	0	0	1	0
13	0.66	0.34	0	0.727	0.273	0	0.347	0.653	0	0	1	0	0.447	0.553	0
14	0.727	0.273	0	0.347	0.653	0	0	1	0	0.447	0.553	0	0.6	0.4	0
15	0.347	0.653	0	0	1	0	0.447	0.553	0	0.6	0.4	0	0.887	0.113	0
16	0	1	0	0.447	0.553	0	0.6	0.4	0	0.887	0.113	0	0.74	0.26	0
17	0.447	0.553	0	0.6	0.4	0	0.887	0.113	0	0.74	0.26	0	0.473	0.527	0
18	0.6	0.4	0	0.887	0.113	0	0.74	0.26	0	0.473	0.527	0	0	1	0
19	0.887	0.113	0	0.74	0.26	0	0.473	0.527	0	0	1	0	0	0.66	0.34
20	0.74	0.26	0	0.473	0.527	0	0	1	0	0	0.66	0.34	0	0.867	0.133
21	0.473	0.527	0	0	1	0	0	0.66	0.34	0	0.867	0.133	0	0.067	0.933
22	0	1	0	0	0.66	0.34	0	0.867	0.133	0	0.067	0.933	0.207	0.793	0
23	0	0.66	0.34	0	0.867	0.133	0	0.067	0.933	0.207	0.793	0	0	1	0
24	0	0.867	0.133	0	0.067	0.933	0.207	0.793	0	0	1	0	0	1	0
25	0	0.067	0.933	0.207	0.793	0	0	1	0	0	1	0	0.153	0.847	0
26	0.207	0.793	0	0	1	0	0	1	0	0.153	0.847	0	0	0.58	0.42
27	0	1	0	0	1	0	0.153	0.847	0	0	0.58	0.42	0.433	0.567	0
28	0	1	0	0.153	0.847	0	0	0.58	0.42	0.433	0.567	0	0.333	0.667	0
29	0.153	0.847	0	0	0.58	0.42	0.433	0.567	0	0.333	0.667	0	0	1	0
30	0	0.58	0.42	0.433	0.567	0	0.333	0.667	0	0	1	0	0.5	0.5	0
31	0.433	0.567	0	0.333	0.667	0	0	1	0	0.5	0.5	0	0	1	0
32	0.333	0.667	0	0	1	0	0.5	0.5	0	0	1	0	0.193	0.807	0
33	0	1	0	0.5	0.5	0	0	1	0	0.193	0.807	0	0	1	0
34	0.5	0.5	0	0	1	0	0.193	0.807	0	0	1	0	0	1	0
35	0	1	0	0.193	0.807	0	0	1	0	0	1	0	1	0	0
36	0.193	0.807	0	0	1	0	0	1	0	1	0	0	1	0	0
37	0	1	0	0	1	0	1	0	0	1	0	0	1	0	0
38	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0
39	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
40	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
41	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0

Table 9: Continued.

Count	47.595	60.227	8.178	47.275	60.547	8.178	46.802	61.02	8.178	47.262	60.56	8.178	47.775	60.047	8.178
Sp	$A_{1.Low}$	$A_{1.Middle}$	$A_{1.High}$	$A_{2.Low}$	$A_{2.Middle}$	$A_{2.High}$	$A_{3.Low}$	$A_{3.Middle}$	$A_{3.High}$	$A_{4.Low}$	$A_{4.Middle}$	$A_{4.High}$	$A_{5.Low}$	$A_{5.Middle}$	$A_{5.High}$
42	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
43	1	0	0	1	0	0	1	0	0	1	0	0	0.667	0.333	0
44	1	0	0	1	0	0	1	0	0	0.667	0.333	0	0.867	0.133	0
45	1	0	0	1	0	0	0.667	0.333	0	0.867	0.133	0	1	0	0
46	1	0	0	0.667	0.333	0	0.867	0.133	0	1	0	0	0.787	0.213	0
47	0.667	0.333	0	0.867	0.133	0	1	0	0	0.787	0.213	0	0.82	0.18	0
48	0.867	0.133	0	1	0	0	0.787	0.213	0	0.82	0.18	0	0.66	0.34	0
49	1	0	0	0.787	0.213	0	0.82	0.18	0	0.66	0.34	0	0.94	0.06	0
50	0.787	0.213	0	0.82	0.18	0	0.66	0.34	0	0.94	0.06	0	0.727	0.273	0
51	0.82	0.18	0	0.66	0.34	0	0.94	0.06	0	0.727	0.273	0	0.727	0.273	0
52	0.66	0.34	0	0.94	0.06	0	0.727	0.273	0	0.727	0.273	0	0.767	0.233	0
53	0.94	0.06	0	0.727	0.273	0	0.727	0.273	0	0.767	0.233	0	0.053	0.947	0
54	0.727	0.273	0	0.727	0.273	0	0.767	0.233	0	0.053	0.947	0	0	0.867	0.133
55	0.727	0.273	0	0.767	0.233	0	0.053	0.947	0	0	0.867	0.133	0	0.467	0.533
56	0.767	0.233	0	0.053	0.947	0	0	0.867	0.133	0	0.467	0.533	0	0.08	0.92
57	0.053	0.947	0	0	0.867	0.133	0	0.467	0.533	0	0.08	0.92	0	0.633	0.367
58	0	0.867	0.133	0	0.467	0.533	0	0.08	0.92	0	0.633	0.367	0.727	0.273	0
59	0	0.467	0.533	0	0.08	0.92	0	0.633	0.367	0.727	0.273	0	0.953	0.047	0
60	0	0.08	0.92	0	0.633	0.367	0.727	0.273	0	0.953	0.047	0	0.907	0.093	0
61	0	0.633	0.367	0.727	0.273	0	0.953	0.047	0	0.907	0.093	0	0.647	0.353	0
62	0.727	0.273	0	0.953	0.047	0	0.907	0.093	0	0.647	0.353	0	0.253	0.747	0
63	0.953	0.047	0	0.907	0.093	0	0.647	0.353	0	0.253	0.747	0	0.78	0.22	0
64	0.907	0.093	0	0.647	0.353	0	0.253	0.747	0	0.78	0.22	0	0.253	0.747	0
65	0.647	0.353	0	0.253	0.747	0	0.78	0.22	0	0.253	0.747	0	0	1	0
66	0.253	0.747	0	0.78	0.22	0	0.253	0.747	0	0	1	0	0	0.827	0.173
67	0.78	0.22	0	0.253	0.747	0	0	1	0	0	0.827	0.173	0	0	1
68	0.253	0.747	0	0	1	0	0	0.827	0.173	0	0	1	0	0.48	0.52
69	0	1	0	0	0.827	0.173	0	0	1	0	0.48	0.52	0.427	0.573	0
70	0	0.827	0.173	0	0	1	0	0.48	0.52	0.427	0.573	0	0	0.887	0.113
71	0	0	1	0	0.48	0.52	0.427	0.573	0	0	0.887	0.113	0.74	0.26	0
72	0	0.48	0.52	0.427	0.573	0	0	0.887	0.113	0.74	0.26	0	0.633	0.367	0
73	0.427	0.573	0	0	0.887	0.113	0.74	0.26	0	0.633	0.367	0	0	1	0
74	0	0.887	0.113	0.74	0.26	0	0.633	0.367	0	0	1	0	0.28	0.72	0
75	0.74	0.26	0	0.633	0.367	0	0	1	0	0.28	0.72	0	0.587	0.413	0
76	0.633	0.367	0	0	1	0	0.28	0.72	0	0.587	0.413	0	0.14	0.86	0
77	0	1	0	0.28	0.72	0	0.587	0.413	0	0.14	0.86	0	0.3	0.7	0
78	0.28	0.72	0	0.587	0.413	0	0.14	0.86	0	0.3	0.7	0	0.313	0.687	0
79	0.587	0.413	0	0.14	0.86	0	0.3	0.7	0	0.313	0.687	0	0.107	0.893	0
80	0.14	0.86	0	0.3	0.7	0	0.313	0.687	0	0.107	0.893	0	0.24	0.76	0
81	0.3	0.7	0	0.313	0.687	0	0.107	0.893	0	0.24	0.76	0	0.707	0.293	0

Table 9: Continued.

Count	47.595	60.227	8.178	47.275	60.547	8.178	46.802	61.02	8.178	47.262	60.56	8.178	47.775	60.047	8.178
Sp	$A_{1.Low}$	$A_{1.Middle}$	$A_{1.High}$	$A_{2.Low}$	$A_{2.Middle}$	$A_{2.High}$	$A_{3.Low}$	$A_{3.Middle}$	$A_{3.High}$	$A_{4.Low}$	$A_{4.Middle}$	$A_{4.High}$	$A_{5.Low}$	$A_{5.Middle}$	$A_{5.High}$
82	0.313	0.687	0	0.107	0.893	0	0.24	0.76	0	0.707	0.293	0	0.433	0.567	0
83	0.107	0.893	0	0.24	0.76	0	0.707	0.293	0	0.433	0.567	0	0.647	0.353	0
84	0.24	0.76	0	0.707	0.293	0	0.433	0.567	0	0.647	0.353	0	0.293	0.707	0
85	0.707	0.293	0	0.433	0.567	0	0.647	0.353	0	0.293	0.707	0	0	0.9	0.1
86	0.433	0.567	0	0.647	0.353	0	0.293	0.707	0	0	0.9	0.1	0.307	0.693	0
87	0.647	0.353	0	0.293	0.707	0	0	0.9	0.1	0.307	0.693	0	0.98	0.02	0
88	0.293	0.707	0	0	0.9	0.1	0.307	0.693	0	0.98	0.02	0	0	0.76	0.24
89	0	0.9	0.1	0.307	0.693	0	0.98	0.02	0	0	0.76	0.24	0	1	0
90	0.307	0.693	0	0.98	0.02	0	0	0.76	0.24	0	1	0	0	0.807	0.193
91	0.98	0.02	0	0	0.76	0.24	0	1	0	0	0.807	0.193	0	0.533	0.467
92	0	0.76	0.24	0	1	0	0	0.807	0.193	0	0.533	0.467	0	0.207	0.793
93	0	1	0	0	0.807	0.193	0	0.533	0.467	0	0.207	0.793	0.133	0.867	0
94	0	0.807	0.193	0	0.533	0.467	0	0.207	0.793	0.133	0.867	0	0.173	0.827	0
95	0	0.533	0.467	0	0.207	0.793	0.133	0.867	0	0.173	0.827	0	0.22	0.78	0
96	0	0.207	0.793	0.133	0.867	0	0.173	0.827	0	0.22	0.78	0	0.413	0.587	0
97	0.133	0.867	0	0.173	0.827	0	0.22	0.78	0	0.413	0.587	0	0.493	0.507	0
98	0.173	0.827	0	0.22	0.78	0	0.413	0.587	0	0.493	0.507	0	0.807	0.193	0
99	0.22	0.78	0	0.413	0.587	0	0.493	0.507	0	0.807	0.193	0	1	0	0
100	0.413	0.587	0	0.493	0.507	0	0.807	0.193	0	1	0	0	0.413	0.587	0
101	0.493	0.507	0	0.807	0.193	0	1	0	0	0.413	0.587	0	0.54	0.46	0
102	0.807	0.193	0	1	0	0	0.413	0.587	0	0.54	0.46	0	0	1	0
103	1	0	0	0.413	0.587	0	0.54	0.46	0	0	1	0	0.033	0.967	0
104	0.413	0.587	0	0.54	0.46	0	0	1	0	0.033	0.967	0	0.087	0.913	0
105	0.54	0.46	0	0	1	0	0.033	0.967	0	0.087	0.913	0	1	0	0
106	0	1	0	0.033	0.967	0	0.087	0.913	0	1	0	0	0.147	0.853	0
107	0.033	0.967	0	0.087	0.913	0	1	0	0	0.147	0.853	0	0.473	0.527	0
108	0.087	0.913	0	1	0	0	0.147	0.853	0	0.473	0.527	0	0.253	0.747	0
109	1	0	0	0.147	0.853	0	0.473	0.527	0	0.253	0.747	0	0.813	0.187	0
110	0.147	0.853	0	0.473	0.527	0	0.253	0.747	0	0.813	0.187	0	0.64	0.36	0
111	0.473	0.527	0	0.253	0.747	0	0.813	0.187	0	0.64	0.36	0	0.667	0.333	0
112	0.253	0.747	0	0.813	0.187	0	0.64	0.36	0	0.667	0.333	0	0.387	0.613	0
113	0.813	0.187	0	0.64	0.36	0	0.667	0.333	0	0.387	0.613	0	0.68	0.32	0
114	0.64	0.36	0	0.667	0.333	0	0.387	0.613	0	0.68	0.32	0	0.527	0.473	0
115	0.667	0.333	0	0.387	0.613	0	0.68	0.32	0	0.527	0.473	0	0.46	0.54	0
116	0.387	0.613	0	0.68	0.32	0	0.527	0.473	0	0.46	0.54	0	0.58	0.42	0

for analysing the change activity of OSS projects. Subject systems were selected from public repositories but selection is biased towards projects with valid *git* repositories.

External validity concerns the generalization of the findings. In the future, we would like to provide more generalized results by considering higher number of OSS projects.

Reliability validity concerns the possibility of replication of the study. The subject systems are available in the public domain. We have attempted to put all the necessary details of the experiment process in the paper.

8. Conclusion and Future Work

The commit activity data available in the development repository of open source software can be used to analyse the evolution of OSS projects as each commit is directly related to the development activity such as code deletion, addition, modification, comments, and file addition. In this study, the Eclipse CDT commits data is analysed to find the regularity and trend in the commit data. The fuzzy data mining algorithm for time series data is used to generate the association rules from the dataset. The dataset is divided into two subsets (training and remaining dataset) to evaluate the pattern of evolution of the Eclipse CDT.

After applying and validating the generated association rule from training dataset, it is found that the rates at which commits performed in training dataset and remaining dataset are different. This thing is again verified by generating the association rules from the remaining set. The generated rules from remaining dataset specify that the data in the considered remaining dataset is more towards lower range of commits performed. This thing validates the applicability of the rules generated from the training dataset.

These association rules indicate that the overall commits in the Eclipse CDT are towards *middle* range except the variation found near the end, where there is a high probability that commits lie in the *lower* range. The continuous availability and existence of commits in the repository of the Eclipse CDT illustrate that the development or evolution of Eclipse CDT is active with most of the commits per month that lie in the *middle* range and at the end lie near to the *lower* range. In the future, we want to consider any prediction algorithm along with the concept of fuzzy data mining algorithm for time series data to give a prediction about the number of commits to be performed in the particular month, although there are other various factors that need to be considered on which the number of commits depends.

Appendix

See Tables 8 and 9.

Competing Interests

The authors declare that they have no competing interests.

References

[1] M. W. Godfrey and Q. Tu, "Evolution in open source software: a case study," in *Proceedings of the IEEE Interantional Conference on Software Maintenance (ICMS '00)*, pp. 131–142, October 2000.

[2] H. Zhang and S. Kim, "Monitoring software quality evolution for defects," *IEEE Software*, vol. 27, no. 4, pp. 58–64, 2010.

[3] Y. Fang and D. Neufeld, "Understanding sustained participation in open source software projects," *Journal of Management Information Systems*, vol. 25, no. 4, pp. 9–50, 2008.

[4] "GIT," July 2015, http://git-scm.com/.

[5] R. Sen, S. S. Singh, and S. Borle, "Open source software success: measures and analysis," *Decision Support Systems*, vol. 52, no. 2, pp. 364–372, 2012.

[6] R. McCleary, R. A. Hay, E. E. Meidinger, and D. McDowall, *Applied Time Series Analysis for the Social Sciences*, Sage, Beverly Hills, Calif, USA, 1980.

[7] R. Grewal, G. L. Lilien, and G. Mallapragada, "Location, location, location: how network embeddedness affects project success in open source systems," *Management Science*, vol. 52, no. 7, pp. 1043–1056, 2006.

[8] I. Herraiz, J. M. Gonzalez-Barahona, G. Robles, and D. M. German, "On the prediction of the evolution of libre software projects," in *Proceedings of the 23rd International Conference on Software Maintenance (ICSM '07)*, pp. 405–414, Paris, France, October 2007.

[9] C. F. Kemerer and S. Slaughter, "An empirical approach to studying software evolution," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 493–509, 1999.

[10] A. Mockus and L. G. Votta, "Identifying reasons for software Changes using historic databases," in *Proceedings of the International Conference on Software Maintenance (ICSM '00)*, pp. 120–130, IEEE, San Jose, Calif, USA, 2000.

[11] C.-H. Chen, T.-P. Hong, and V. S. Tseng, "Fuzzy data mining for time-series data," *Applied Soft Computing Journal*, vol. 12, no. 1, pp. 536–542, 2012.

[12] C. C. H. Yuen, "An empirical approach to the study of errors in large software under maintenance," in *Proceedings of the IEEE International Conference on Software Maintenance (ICSM '85)*, pp. 96–105, 1985.

[13] C. C. H. Yuen, "A statistical rationale for evolution dynamics concepts," in *Proceedings of the IEEE International Conference on Software Maintenance (ICSM '87)*, pp. 156–164, September 1987.

[14] C. C. H. Yuen, "On analytic maintenance process data at the global and the detailed levels: a case study," in *Proceedings of the International Conference on Software Maintenance (ICSM '88)*, pp. 248–255, IEEE, Scottsdale, Ariz, USA, 1988.

[15] M. Goulão, N. Fonte, M. Wermelinger, and F. B. Abreu, "Software evolution prediction using seasonal time analysis: a comparative study," in *Proceedings of the 16th European Conference on Software Maintenance and Reengineering (CSMR '12)*, pp. 213–222, IEEE, Szeged, Hungary, March 2012.

[16] B. Kenmei, G. Antoniol, and M. Di Penta, "Trend analysis and issue prediction in large-scale open source systems," in *Proceedings of the 12th IEEE European Conference on Software Maintenance and Reengineering (CSMR '08)*, pp. 73–82, Athens, Greece, April 2008.

[17] F. Caprio, G. Casazza, U. Penta, M. D. Penta, and U. Villano, "Measuring and predicting the Linux kernel evolution," in *Proceedings of the International Workshop of Empirical Studies on Software Maintenance*, Florence, Italy, November 2001.

[18] E. Fuentetaja and D. J. Bagert, "Software evolution from a time-series perspective," in *Proceedings of the IEEE International Conference on Software Maintenance*, pp. 226–229, October 2002.

[19] M. Kläs, F. Elberzhager, J. Münch, K. Hartjes, and O. Von Graevemeyer, "Transparent combination of expert and measurement data for defect prediction—An Industrial Case Study," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE '10)*, pp. 119–128, Cape Town, South Africa, May 2010.

[20] U. Raja, D. P. Hale, and J. E. Hale, "Modeling software evolution defects: a time series approach," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 21, no. 1, pp. 49–71, 2009.

[21] G. Antoniol, G. Casazza, M. D. Penta, and E. Merlo, "Modeling clones evolution through time series," in *Proceedings of the IEEE International Conference on Software Maintenance (ICSM '01)*, pp. 273–280, IEEE Computer Society, Florence, Italy, November 2001.

[22] L. Yu, "Indirectly predicting the maintenance effort of open-source software," *Journal of Software Maintenance and Evolution*, vol. 18, no. 5, pp. 311–332, 2006.

[23] A. Amin, L. Grunske, and A. Colman, "An approach to software reliability prediction based on time series modeling," *Journal of Systems and Software*, vol. 86, no. 7, pp. 1923–1932, 2013.

[24] Forecasting Model, http://people.duke.edu/~rnau/411fcst.htm.

[25] T.-P. Hong, C.-S. Kuo, and C.-C. Chi, "A fuzzy data mining algorithm for quantitative values," in *Proceedings of the 3rd International Conference on Knowledge-Based Intelligent Information Engineering Systems (KES '99)*, pp. 480–483, September 1999.

[26] H. Suresh and K. Raimond, "Mining association rules from time series data using hybrid approaches," *Proceedings of International Journal of Computational Engineering Research*, vol. 3, no. 3, pp. 181–189, 2013.

[27] "GIT HUB," July 2015, https://github.com/.

[28] "Eclipse", "Where did Eclipse come from?", Eclipse Wiki.

[29] "Eclipse," 2015, https://eclipse.org/cdt/.

[30] "Eclipse," July 2015, http://www.eclipse.org/cdt/doc/overview/#/3.

[31] L. A. Zadeh, "Fuzzy sets," *Information and Computation*, vol. 8, no. 3, pp. 338–353, 1965.

[32] J. Han and M. Kamber, *Data Mining Concepts and Techniques*, Morgan Kaufman Academic Press, 2001.

[33] R. Agrawal, "Mining association rules between sets of items in large databases," in *Proceedings of the ACM SIGMOD Conference*, pp. 207–216, Washington, DC, USA, May 1993.