Software Defect Association and Correction Effort Prediction

By

A.P.Arulanand
Reg. No. 71204621002

Of

KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE

A PROJECT REPORT
Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements
for the award of the degree
of

MASTER OF COMPUTER APPLICATIONS

July, 2007

Certificate

Kumaraguru College of Technology

Coimbatore – 641006.

Department of Computer Applications

Bonafide Certificate

Certified that this project report titled **Software Defect Association and Correction Effort Prediction** is the bonafide work of **Mr. A.P.Arulanand** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Project Guide

Head of the Department

Submitted for the University Examination held on 02/07/2007

Internal Examiner

External Examiner 02/07/07

Company Certificate

ADVENTURE TECHNOLOGIES

Date: 28-May-2007

PROJECT COMPLETION

Sub.: Student Project Completion Reg.

This letter is issued to confirm that the following student of **Kumaraguru College of Technology, Coimbatore** has successfully completed his final year project "**Software Defect Association and Correction Effort Prediction**" in our concern from the period of 25th December 2006 to 25th May 2007. The performance of the student during the term is good.

Name	: Arulanand A P
Roll No.	: 04MCA02
Course	: M.C.A.

ADVENTURE TECHNOLOGIES
112, Jubilee Complex
Chinnasamy Naidu Road
New Siddhapudur, Coimbatore-641 044

112, Jubilee Complex
Chinnasamy Naidu Road
New Siddhapudur
Coimbatore - 641 044
Phone : 0422 - 5387455
E-mail : adventure_tech@touchtelindia.net

94439 33189 / 98432 38246

Abstract

ABSTRACT

The project titled "**Software Defect Asssociation and Correction Effort Prediction**" addresses association rule mining based methods to predict defect associations and defect correction effort.

This project helps the developers to detect software defects and assist project managers in allocating testing resources more effectively. The success of a software system depends not only on cost and schedule, but also on quality. The prediction of software defects, i.e., deviations from specifications or expectations which might lead to failures in operation, has been an important research topic in the field of software engineering. Clearly, they are a proxy for reliability, but, unfortunately, reliability is extremely difficult to assess prior to full deployment.

Acknowledgement

Acknowledgement

I express my grateful thanks to our beloved principal, **Dr. Joseph V Thanikal**, Kumaraguru College of Technology, Coimbatore, for giving me an opportunity to take up this project.

I express my deep sense of gratitude to **Dr.M.Gururajan,** Professor and Head of Department of Computer Applications for extending their help in providing all the facilities at college for the successful completion of the project.

I would like to express my deepest and sincere gratitude to **Mr. Ganesh Babu**, Lecturer, Department of Computer Applications, for his guidance, support, cooperation and valuable suggestions during the course of this project.

I also thank **Mr.A.Muthukumar,** Assistant Professor and Course Coordinator, Computer applications for providing his valuable suggestions and encouragement throughout my project.

Contents

Table of Contents

List of Tables

List of Figures

Introduction

CHATER 1

INTRODUCTION

1.1 SYSTEM OVERVIEW

The project titled "Software Defect Asssociation and Correction Effort prediction" addresses association rule mining based methods to predict defect associations and defect correction effort.

This project helps the developers to detect software defects and assist project managers in allocating testing resources more effectively. The success of a software system depends not only on cost and schedule,but also on quality. The prediction of software defects, i.e., deviations from specifications or expectations which might lead to failures in operation , has been an important research topic in the field of software engineering. Clearly, they are a proxy for reliability, but, unfortunately, reliability is extremely difficult to assess prior to full deployment.

The goal of this project is to use defect type data to predict software defect associations that are the relations among different defect types and perform correction effort prediction. The defect associations can be used for three purposes:

- find as many related defects as possible to the detected defect(s) and, consequently, make more-effective corrections to the software
- to help evaluate reviewers results during an inspection
- to assist managers in improving the software process through analysis of the reasons some defects frequently occur together.

If the analysis leads to the identification of a process problem, managers have to come up with a corrective action. for each of the associated defects, here also predict the likely effort required to isolate and correct it. This can be used to help project managers improve control of project schedules. Both defect association prediction and defect correction effort prediction methods are based on the association rule mining method. Although association rule mining aims to discover the co-occurring patterns of the attributes in databases, it has been shown that association rule mining based classification, associative classification, frequently has higher classification accuracy than other classification methods. The underlying reason is that these methods use "heuristic/ greedy "search techniques to build classifiers, they induce a representative subset of rules.

1.2 COMPANY PROFILE

Founded in 2004, Adventure Technologies is a privately funded, solution, products and services provider in Internet and Distributed Computing. The strengths of Adventure Technologies are five fold: Technology, Integration Skills, Management & Development disciplines, Broad Based market experience and Off Shore Resources to give you cost advantage. The advantage and effectiveness of Adventure Technologies is enhanced by the structured approach to project management and all the development processes. Further, the company has accumulated a depth of system development and Internet application across a wide spectrum of markets that include for example: HIPAA, Business Process and Workflow Integration, Supply Chain B2B, Airport Display Systems, Smart Card based Toll Collection & Management Systems, Payment Gateways, Equity Charting Systems and Embedded Systems.

Adventure's management strength is embodied in its executives who bring product development experience, remote product development methodology and practices to the table. It is adequately funded to support growth in infrastructure and new development.

Adventure Technologies is aimed at enabling enterprises achieve the maximum ROI from their investments in IT solutions. Adventure Technologies provides End-to-End Solutions from implementing Custom Software Development and Packaged Software such as ERP and Business Intelligence Solutions to Application Management services for all types of solutions.

The company has successfully leveraged its expertise in Offshore Outsourcing to set up Dedicated Offshore Development Center that provides Application Management services including Help Desk Support for Software Product Companies.

The Adventure technology includes the following services,

- Application Development
- Application Management
- Verification and Validation
- Packaged Software
- Business Inteliigence

System Study and Analysis

CHAPTER 2

SYSTEM STUDY AND ANALYSIS

2.1 PROBLEM STATEMENT

Data mining technology has emerged as a means of identifying patterns and trends from the large quantity of Data. Much current software defect prediction focuses only on the number of defects remaining in the software system. They don't determine whether defects accompanied with other defects to take an defect correction effort prediction. This project states the problem of defect prediction in this scenario.

The goal of the project is to identify the defects accompanied with other defects to predict the correction effort. That helps the developers to detect software defects and helps project managers to allocate test resources more effectively.

2.2 EXISTING SYSTEM

Current defect prediction work focuses on estimating the number of defects remaining in software systems with code metrics, inspection data, and process-quality data by statistical approaches capture-recapture (CR) models and detection profile methods (DPM).

The prediction result, which is the number of defects remaining in a software system, can be used as an important measure for the software developer [16], and can be used to control the software process and gauge the likely delivered quality of a software system.

2.2.1 Drawbacks of the Existing System

The drawbacks of the existing system can be summarized as:

- Estimates defects remaining in the software system and wont determine the associated defects with identified defects.
- Takes no correction prediction of defects and it just controls the software process.

2.3 PROPOSED SYSTEM

The proposed system works to the predictions of defect associations and corresponding defect correction effort. NASA SEL dataset is preprocessed for this project. Association rule mining based method is used to discover defect associations and the patterns between a defect and the corresponding defect correction effort.

2.3.1 Advantages of the Proposed System

The expected benefits of proposed system are as:

- Determine defects and other defects associated with the identified defects.
- Predict the effort needed to take defect correction.
- Defect correction prediction helps managers to allocate testing resource effectively.

2.4 FEASIBILITY ANALYSIS

Feasibility analysis is the measure of how beneficial or practical the development of Information System will be to the Organization. Once the problem is explained information is gathered about the system to test whether the system is viable Technically, Financially and Operationally. Thus, feasibility study is carried out in three phases as follows:

2.4.1 Technical Feasibility

Technical Feasibility is the measure of practicality of a specific technical solution and the availability of technical resources and expertise. It centers on the existing computer system (hardware, software, etc.) and to what extent it can support the new addition.

The proposed system is to be developed using Java, MS-Access, which are some of the leading technologies in the market. These technologies work on all architectures i.e. on all available platforms. Hence if the system is to be executed on Linux platform later, the system can be ported across to it. This system works on any back-end (Oracle 9i, MS-Access). These features of the selected technologies are quite beneficial to the proper functioning of the system in different environments.

2.4.2 Operational Feasibility

Operational Feasibility asks if the system will work when it is developed and installed. It checks for the support of the management, the current business methods, user's involvement and their attitude towards the proposed system, etc.

The proposed system can be applied in any knowledge discovering system requiring secured association rule mining. It supports in fining defects and correction effort needed for any input of data.

2.4.3 Economic Feasibility

Economic Feasibility is the measure of the cost-effectiveness of the proposed system. The investment to be made in the proposed system must prove a good investment to the organization by returning benefits equal to or exceeding the costs incurred in developing the system.

The proposed benefits of the system will outweigh the costs to be incurred during system developed since the system does not require procurement of additional hardware facilities it is economically feasible. No extra cost is involved for because of the deployment of simple apriori algorithm and association rule mining. Porting the system to work over any Back-end database can be done without increase in cost.

2.5 APPLICATIONS OF THE SYSTEM

The application area for this concept Software Defect Association and Correction Effort Prediction is "Software Development Industry" to help developer to detect software defect and project managers to control software process and allocate testing resources effectively.

Development Environment

CHAPTER 3

DEVELOPMENT ENVIRONMENT

3.1 HARDWARE REQUIREMENTS

The hardware support required for deploying the application is:-

Processor	: Pentium 3 Processor or above/Athlon Processor:
Hard Disk	: 20GB or more

3.2 SOFTWARE REQUIREMENTS

The software support required for deployment is:-

Architectural Support	: JDBC
Operating System	: Windows XP
Database	: MS-Access/Oracle9i
Software for Development	: Java

3.3 PROGRAMMING ENVIRONMENT

3.3.1 JDBC

The Java Database Connectivity Application Programming Interface (API) is an API currently being designed by Sun Microsystems that provides a Java language interface to the X/Open SQL Call Level Interface standard. This standard provides a DBMS-independent interface to relational databases that defines a generic SQL database access framework. The most visible implementation of the X/Open SQL CLI is Microsoft's ODBC (Open Database Connectivity). This API defines a common SQL syntax and function calls that can be used by developers to send SQL commands to and retrieve data from SQL databases. ODBC-enabled applications make use of database drivers (similar in concept to other device drivers) installed on the system that allow applications to talk to a vendor's database. Using this methodology, all of the DBMS-specific code is placed inside the ODBC driver and the application developer is shielded from implementation-specific problems in theory. Practically speaking, it is sometimes difficult to completely remove vendor-specific syntax from all ODBC operations, but in most cases, it is a relatively simple task to port ODBC to run on a new database server.

3.3.1.1 JDBC overview

The JDBC API is expressed as a series of abstract Java interfaces within the java.sql package that will be provided as part of the JDK 1.1 release. Here are the most commonly used interfaces:

- **java.sql.DriverManager:**-Manages the loading and unloading of database drivers from the underlying system.
- **java.sql.Connection:**-Handles the connection to a specific database
- **java.sql.Statement:**-Contains an SQL statement to be passed to the database; two subtypes in this interface are the PreparedStatement

(for executing a precompiled SQL statement) and the CallableStatement (for executing a database stored procedure).

- **java.sql.ResultSet:**-Contains the record result set from the SQL statement passed to the database

JDBC-enabled applets and applications make use of database drivers to connect to remote databases. What sets JDBC apart from ODBC is that these drivers can actually be applets themselves that get uploaded to the client system at runtime. Therefore, the overall Java model of a "thin client" querying a powerful database remains.

3.3.2 Microsoft Access

Access is a versatile application for creating databases. Although it may not be as powerful as industrial strength database management systems, Access is very popular due to the vast number of features it provides. Many businesses also favor Access because it is integrated with all the other Microsoft Office products.

As with all of the Microsoft Office products, there have been numerous updates to Access over the years. The most recent version of Access, known as Access 2000, was made available to the public in June of 1999. New features of this version include easy organizing and sharing of work over intranets and networks, integrating databases with the World Wide Web, interface improvements, and many more.

3.3.3 Oracle 9i

Oracle Corporation strives to comply with industry-accepted standards and participates actively in SQL standards committees. The strengths of SQL provide benefits for all types of users, including application programmers, database administrators, managers, and end users. Technically speaking, SQL is

a data sublanguage. The purpose of SQL is to provide an interface to a relational database such as Oracle, and all SQL statements are instructions to the database.

3.3.3.1 Features of Oracle 9i

ORACLE 9i provides statements for a variety of tasks, including:

- Querying data
- Inserting, updating, and deleting rows in a table
- Creating, replacing, altering, and dropping objects
- Controlling access to the database and its objects
- Guaranteeing database consistency and integrity
- Supports PL/SQL

3.3.4 JAVA

Java is a programming language expressly designed for use in the distributed environment of the Internet. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build a small application module or applet for use as part of a Web page. Applets make it possible for a Web page user to interact with the page.

The major characteristics of Java are:

- The programs you create are portable in a network. Your source program is compiled into what Java calls bytecode, which can be run anywhere in a network on a server or client that has a Java virtual machine. The Java virtual machine interprets the bytecode into code that will run on the real computer hardware. This means that individual

computer platform differences such as instruction lengths can be recognized and accommodated locally just as the program is being executed. Platform-specific versions of your program are no longer needed.

- The code is robust, here meaning that, unlike programs written in C++ and perhaps some other languages, the Java objects can contain no references to data external to themselves or other known objects. This ensures that an instruction can not contain the address of data storage in another application or in the operating system itself, either of which would cause the program and perhaps the operating system itself to terminate or "crash." The Java virtual machine makes a number of checks on each object to ensure integrity.
- Java is object-oriented, which means that, among other characteristics, an object can take advantage of being part of a class of objects and inherit code that is common to the class. Objects are thought of as "nouns" that a user might relate to rather than the traditional procedural "verbs." A method can be thought of as one of the object's capabilities or behaviors.
- In addition to being executed at the client rather than the server, a Java applet has other characteristics designed to make it run fast.

System Design and Development

CHAPTER 4

SYSTEM DESIGN AND DEVELOPMENT

4.1 ELEMENTS OF DESIGN

System Design is the most creative and challenging phase in the development of a software system. Design implies to a description of the final system and the process by which it is developed. The first step is to determine what input data is needed for the system and then to design a database that will meet the requirements of the proposed system. The next step is to determine what outputs are needed from the system and the format of the output to be produced.

During the design of the proposed system some areas where attention is required are:

- What are the inputs required and the outputs produced?
- How should the data be organized?
- What will be the processes involved in the system?
- How should the screen look?

The steps carried out in the design phase are as follows:

- Modular Design
- Output Design
- Database Design

4.1.1 Modular Design

It is always difficult for any System Development team to grasp a system without breaking it into several smaller systems. These smaller systems will be a part of the original system yet they will be independent in the sense that they will incorporate within them the major functionalities of the proposed system.

A software system is always divided into several subsystems which make it easier to develop and perform tests on the whole system. The subsystems are known as the modules and the process of dividing an entire system into subsystems is known as Decomposition.

The modules identified for the proposed system are as below:

- Defect Data Collection and DataBase Management
- Basic Association Rule Mining
- Defect Association Prediction
- Defect Correction Effort Prediction

4.1.1.1 Defect Data Collection and Database Management

Perform data preparation and basic database management functions. The database is of type transactional database where each record would have variable number of columns. We use two data structures, namely Itemset and ItemsetCollection to carry out the database operations.

An Itemset is a list of items. Each item is stored as a string. So, an Itemset is an array of strings. For example, {bread butter milk} is an Itemset containing "bread", "butter" and "milk" as items. An Itemset is also called as a transaction. A transaction refers to an action that generates a list of items. For example, in a

shopping complex, sales bills are generated daily containing a list of items that are sold. Each sales bill is a transaction, containing the list of items that are sold.

An ItemsetCollection is a list of Itemsets. Each element is an object of Itemset class. So the member variable of this class is an array of objects of Itemsets. An ItemsetCollection can also be called as a database. For example, a set of sales bills generated in a year contributes to an ItemsetCollection.

The association rule mining generates all association rule that have a support greater than the minimum support *min.supp*(A=>C) and the rule must also have confidence greater than minimum confidence *min.conf*(A=>C).(Let A and C are set with items).

4.1.1.2 Basic Association Rule Mining

The first step in the generation of association rules is the identification of large itemsets. A number of algorithms have been proposed in the literature for obtaining large itemsets. A commonly used algorithm for this purpose is the Apriori algorithm.

Algorithm Apriori relies on the following subset principle: Every nonempty subset of a large itemset must itself be a large itemset. The algorithm uses this principle in a bottom-up manner. Assume that a support threshold of a is used. Let Li denote the collection of large itemsets where each itemset has i items. The algorithm begins by identifying all the sets in L1. This can be done in a straightforward manner by counting the number of transactions in which each item appears, and retaining only those items that appear in at least a transactions. Each item that has the necessary support forms a singleton large itemset and is included in L1. By the subset principle, items that appear in less than a transactions cannot be part of any large itemset; therefore, they are dropped from further consideration. The collection L2 can be constructed by

considering each pair of sets in L1 and retaining only those pairs that appears in at least a transaction. In general, having constructed Li, the collection Li+1 is constructed by considering pairs of sets, one from Li and another from L1 and eliminating those for which the support is smaller than a. This procedure is continued until all large itemsets up to the desired maximum size have been obtained.

4.1.1.2.1 Rule-Ranking Strategy

Before prediction, we rank the discovered rules according to the length-first strategy. The length-first strategy was used for two reasons. First, for the defect association prediction, the length-first strategy enables us to find out as many defects as possible that coincide with known defect(s), thus preventing errors due to incomplete discovery of defect associations. Second, for the defect correction effort prediction, the length-first strategy enables us to obtain more-accurate rules, thus improving the effort prediction accuracy.

4.1.1.3 Defect Association Prediction

Here, we discover software defect associations from historical software engineering data set, and help determine whether or not a defect(s) is accompanied by other defect(s).

The next step is to predict whether or not a k-defect will occur with others. Then, for the k-defect, we scan the rules one by one and identify the rule whose antecedent contains the k-defect. After that, we merge the consequent of the corresponding rule with the k-defect and generate a (k + 1)-defect. For the (k + 1)-defect, we repeat the process until there are no rules available, the {(k + n)-defect} ? {k-defect} is the defect(s) which occurred with the k-defect.

4.1.1.4 Defect Correction Effort Prediction

Attempts to determine what these defects are and how much effort might be expected to be used when we correct them.

Specifically, the procedure of association rule mining was adapted as follows:

1. Compute the frequent item sets that occur together in the training data set at least as frequently as a predetermined minimum support. The item sets mined must also contain the effort labels.
2. Generate association rules from the frequent item sets, where the consequent of the rules is the effort. In addition to the minimum support threshold, these rules must also satisfy a minimum confidence minimum confidence.

The resultant rules are evaluated using the following measures: prediction accuracy, false-negative rate, and false-positive rate. We also explore the impact of support and confidence levels on prediction accuracy, false negative rate, false positive rate, and the number of rules as well. We find that higher support and confidence levels may not result in higher prediction accuracy, and a sufficient number of rules is a precondition for high prediction accuracy.

4.1.2 Output Design

Reports are generated as output for the users to view and take print-outs. Different reports are generated for different criteria. The reports present in the system are:

- Itemset list report*
- Association Rule obtained

Itemset list report produces a list of itemset which has support count exceeding the minimum threshold value. Support values(10,20 and 30), Confidence values(40,50, and 60).

4.1.3 Database Design

The data we used is SEL Data which is a subset of the online database created by the NASA/GSFC Software Engineering Laboratory (SEL). The subset includes defect data of more than 200 projects completed over more than 15 years. The SEL Data is a database consisting of 15 tables that provide data on the projects' software characteristics (over all and at a component level), changes and errors during all phases of development, and effort and computer resources used.

We use SQL to extract defect data from different tables of the SEL data and obtained the basic defect data set. The defect data is very simple, and consists of defect types and the corresponding dates on which the need for change was determined.

Input: *Rules* – the defect association rules generated by the association rule mining Apriori algorithm.
Output: *Rules* – the ranked defect association rules by applying the proposed rule ranking length-first strategy, the most priori first.

```
1: for each rule r_i ∈ Rules do
2:   for each rule r_{j>i} ∈ Rules do
3:     if Length(r_i) < Length(r_j) then
4:       r_i ↔ r_j; // make more priori first
5:     else if Length(r_i) ≡ Length(r_j) then
6:       if Conf(r_i) < Conf(r_j) then
       // the priority depends on confidence values for rules with the same length
7:         r_i ↔ r_j;
8:       else if Conf(r_i) ≡ Conf(r_j) then
9:         if Supp(r_i) < Supp(r_j) then
           // the priority depends on support values for rules with the same length
           // and the same confidence value
10:          r_i ↔ r_j;
11:        else if Supp(r_i) ≡ Supp(r_j) then
12:          if AlphabetOrder(r_i) > AlphabetOrder(r_j) then
13:            r_i ↔ r_j;
14:          end if
15:        end if
16:      end if
17:    end if
18:    r_i ≺ r_j; // r_i has higher priority
19:  end for
20: end for
```

RULE-RANKING STRATERGY

4.2 Table Structure

Table No : 4.2.1

Table Name: PROJECT

FIELD NAME	TYPE
PROJECT_NAME	VARCHAR2(10)
PROJECT_ALIAS	VARCHAR2(5)
APPLICATION_DOMAIN	VARCAHR2(10)
DEVELOPMENT_STATUS	VARCHAR2(10)
MAINTENANCE_STATUS	VARCHAR2(10)
ORGANIZATION	VARCHAR2(10)
TYPE	VARCHAR2(5)
OLD_PROJECT_TYPE	VARCHAR2(5)
OLD_PROJECT_STATUS	VARCHAR2(5)
START_DAT	DATE
END_DATE	DATE

Table No :4.2.2

Table Name: APPLICATION_BUILD

FIELD_NAME	TYPE
APPLICATION_ID	NUMBER(5)
FORM_ID	NUMBER(5)
BUILD_NUMBER	NUMBER(5)

Table No :4.2.3

Table Name: APPLICATION_PROGRAM

FIELD_NAME	TYPE
APPLICATION_ID	NUMBER(5)
APPLICATION_PROGRAM_NAME	VARCHAR2(10)
FUNCTION	VARCHAR2(10)
DESCRIPTION	VARCHAR2(10)
SUBSYSTEM_PREFIX	VARCHAR2(10)

Table No : 4.2.4

Table Name: BUILD

FIELD_NAME	TYPE
FORM_ID	NUMBER(5)
BUILD_NUMBER	NUMBER(5)
ESTIMATE_OR_ACTUAL	NUMBER(5)
START_DATE	DATE
END_DATE	DATE
TYPE	VARCHAR2(5)

Table No : 4.2.5

Table Name: COMPONENT_CHANGE

FIELD_NAME	TYPE
CHANGE_ID	NUMBER(5)
COMPONENT_ID	NUMBER(5)

Table no :4.2.6

Table Name: PROGRAM_COMPONENT

FIELD_NAME	TYPE
COMPONENT_ID	NUMBER(5)
APPLICATION_ID	NUMBER(5)
BUILD_NUMBER	NUMBER(5)

Table No : 4.2.7

Table Name: COMPONENT

FIELD_NAME	TYPE
COMPONENT_ID	NUMBER(5)
CLASS_ID	NUMBER(5)
ORIGIN	VARCHAR2(10)
CONFIGURE_DATE	DATE
ORIGIN_PROCESS	VARCHAR2(10)
TYPE	VARCHAR2(5)
LANGUAGE	VARCHAR2(10)
REUSE_MODIFICATION	CHAR(2)
APPROVED_BY	VARCHAR2(15)
LOCATION	VARCHAR2(15)
OLD_COF_TYPE	VARCHAR2(10)
ORI_TYPE	VARCHAR2(10)
ORI_DIFFICULT	VARCHAR2(10)
FINAL_ORIGIN	VARCHAR2(10)

Table No : 4.2.8

Table Name: PRODUCT_RELEASE

FIELD_NAME	TYPE
PROJECT_NAME	VARCHAR2(15)
PRODUCT_NAME	VARCHAR2(15)
RELEASE_NUMBER	NUMBER(5)
VERSION_NUMBER	NUMBER(5)
PRODUCT_DESCRIPTION	VARCHAR2(25)
RELEASE_CLASS	VARCHAR2(10)
RELEASE_TYPE	VARCHAR2(10)
START_DAT	DATE
END_DATE	DATE

Table No : 4.2.9

Table Name : PROJECT_ESTIMATE

FIELD_NAME	TYPE
FORM_ID	NUMBER(5)
TOTAL_SUBSYSTEMS	NUMBER(5)
TOTAL_COMPONENTS	NUMBER(5)
TOTAL_LINE	NUMBER(5)
TOTAL_NEW_LINE	NUMBER(5)
TOTAL_MOD_LINE	NUMBER(5)
TOTAL_OLD_LINE	NUMBER(5)
PROGRAMMER_HR	NUMBER(5)
MANAGEMENT_HR	NUMBER(5)
SERVICES_HR	NUMBER(5)
NEW_COMPONENT	NUMBER(5)
MODIFIED_COMPONENT	NUMBER(5)
REUSED_COMPONENT	NUMBER(5)

Table No : 4.2.10

Table Name : CHANGE

FIELD_NAME	TYPE
CHANGE_ID	NUMBER(5)
BUILD_NUMBER	NUMBER(5)
SPR_STR_SCR	NUMBER(5)
EFFORT_TO_ISOLATION	VARCHAR2(15)
EFFORT_TO_COMPLETE	VARCHAR2(15)
COMMUNICATION	CHAR(2)
DATE_DETERMINED	DATE
COMPLETION_DATE	DATE
COMPONENTS_EXAMINED	NUMBER(5)
TYPE	VARCHAR2(10)
APPROVED_BY	VARCHAR2(15)
DESCRIPTION	VARCHAR2(25)
LOCATION	VARCHAR2(10)
OLD_CRF	CHAR(2)
EFF_ADA	CHAR(2)
EFF_ONE	CHAR(2)
EFF_OTHER	CHAR(2)

4.3 DATA FLOW DIAGRAM

Data flow diagrams are graphical representation depicting information regarding the flow of control and the transformation of data from input to output. The DFD may be used to represent the system or software at any level of abstraction. In fact, DFD can be partitioned into levels. A Level 0 DFD called Context Level Diagram represents the entire software system as a single bubble with its interactions.

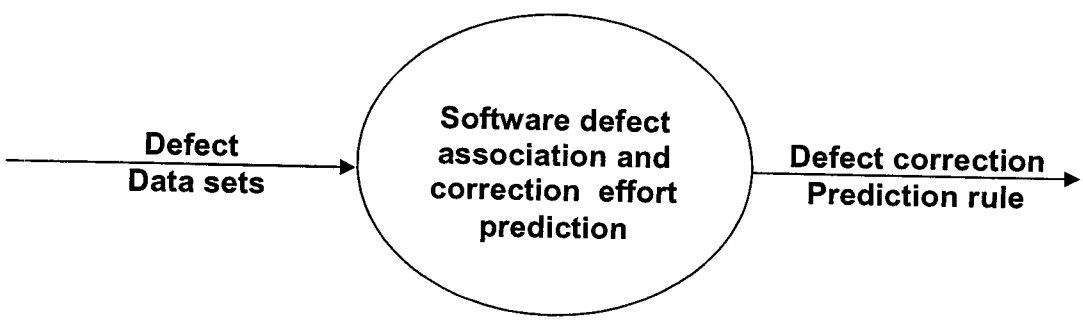


Figure 4.3.1: Context Flow Diagram

Level 1 DFD

It will explain the major modules in the whole system, i.e., how the data flow between each of these modules. The interaction of each process is shown.

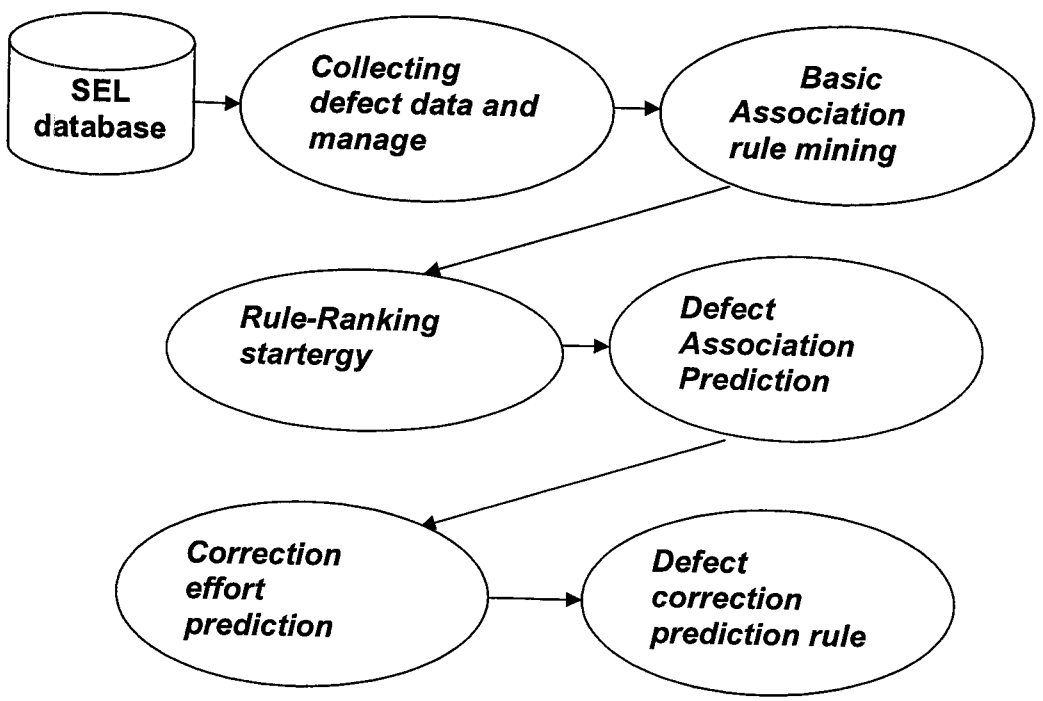


Figure 4.3.2: Level 1 DFD

Implementation

CHAPTER 5

SYSTEM IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION OVERVIEW

Implementation includes all those activities that take place to convert from the old system to the new. The new system may be totally new, replacing an existing system.

In this project as explained in the system stage the Software defect association and correction effort prediction is implemented in JAVA. The association rule is produced for the defect data set. The association rule is ranked by first-length technique that helps to predict defect association.

By repeating the rule accompanied defects are obtained with defects identified and the defect correction prediction rule is generated. Test cases are performed and its results are matching with expected results.

5.2 SOFTWARE TESTING

Testing is a critical element of software quality and assurance and represents the ultimate review of specification design and coding. It is a vital activity that has to be enforced in the development of any system. This could be done in parallel during all the phases of system development. The feedback received from these tests can be used for further enhancement of the system under consideration. The testing phase conducts test using the Software Requirement Specification as a reference and with the goal to see whether the system satisfies the specified requirements.

Test cases are generated for each screen. These test cases will cover every possibility which could result in both positive and negative results. These test plans are maintained for any further testing done on the system. The test plan stores information such as, the test script/input, expected output, actual output, comments and the name of the tester. This plan will be followed for all types of testing done in the system.

The main types of tests carried out on this project are:

- Unit Test
- Integration Test
- System Test

5.2.1 Unit Testing

Module or Unit Testing is the process of testing all the program units that make up a system. Unit testing focuses on an individual module thus allowing one to uncover all the errors made logically and while coding in the module.

In this project, each process is tested separately as a unit. Initially the flow of control and data passing through each process is checked. When considering a module as a unit, the flow of data and control through the whole module is tested. The result is stored in the test plan. In a process, each control is further tested in unit testing. Once the errors are rectified, the testing procedure is repeated with same test cases to ensure this hasn't produced new errors. Hence this is a continuous process.

Test cases were generated to test the control flow of each unit or module. Almost all cases needed for testing control flows have been generated.

Test Cases for Mining Screen:

S.NO	Test Case	Expected Result	Observed Result	Status
1.	Sup.threshold:20 Conf.threshold:40	Mining Process	Mining Process	Pass
2.	Sup.threshold:40 Conf.threshold:20	Invalid threshold value	Invalid threshold value	Pass
3.	Sup.threshold:30 Conf.threshold:50	Mining Process	Mining Process	Pass

5.2.2 Integration Testing

Integration testing tests the process of integrating the various modules to form the completed system. Integration starts with a set of units each individually tested in isolation and ends when the entire application has been built. Integration testing verifies that the combined units function together correctly. It facilitates in finding problem that occur at interface or communication between the individual parts.

This system follows top-down integration testing. Modules were linked to the main menu in a sequence as required in the real time operating mode of the system.

5.3.3 System Testing

System testing is actually a serious of different tests, whose primary purpose is to fully exercise the computer-based system. This helps in verifying that all the system elements have been properly integrated and perform the allocated functions. It verifies the entire product after having integrated all software and hardware components, and validates it according to the original project requirement. The system testing takes into consideration the hardware, and the software. The proposed system is able to be run on any back-end database. The project is tested against recovery from errors.

Conclusion

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

An application of association rule mining to predict software defect associations and defect correction effort with SEL defect data has been presented. This is important in order to help developers detect software defects and project managers improve software control and allocate their testing resources effectively.

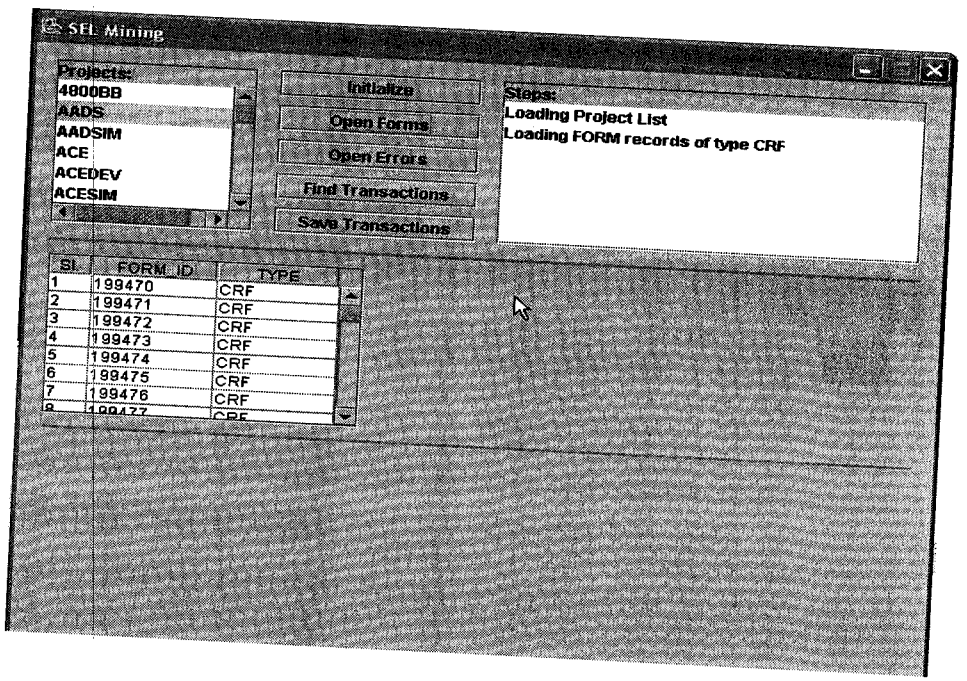
6.2 FUTURE ENHANCEMENTS

In future the system can be improved to make better correction prediction by using improved algorithms. This system supports any database as its back-end. Continued research can improve the system to predict better correction effort for the managers. According to the development of technology the system can be upgraded to its need. This helps to detect defects accompanied with the identified defects to predict correction effort in better manner for the newly found defects in the database.

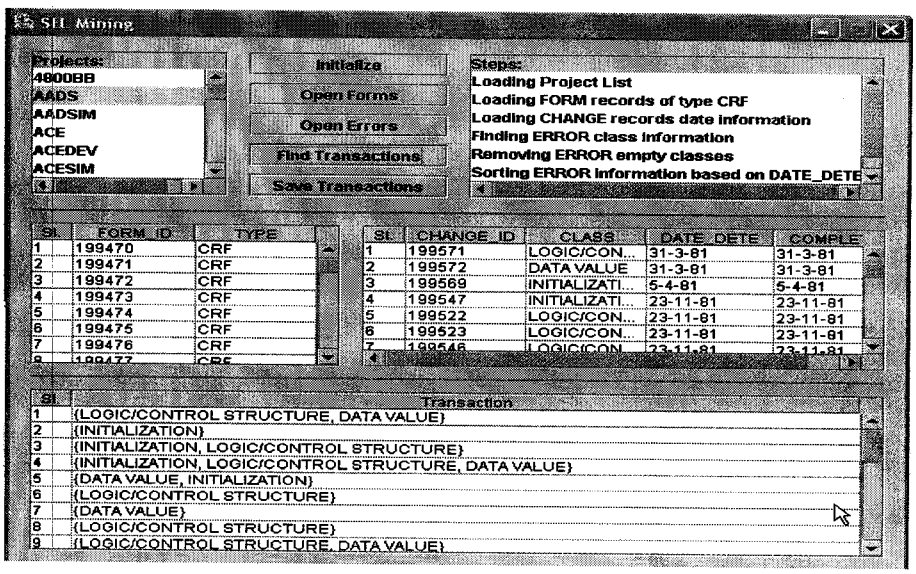
Appendices

APPENDICES

Open FORMS

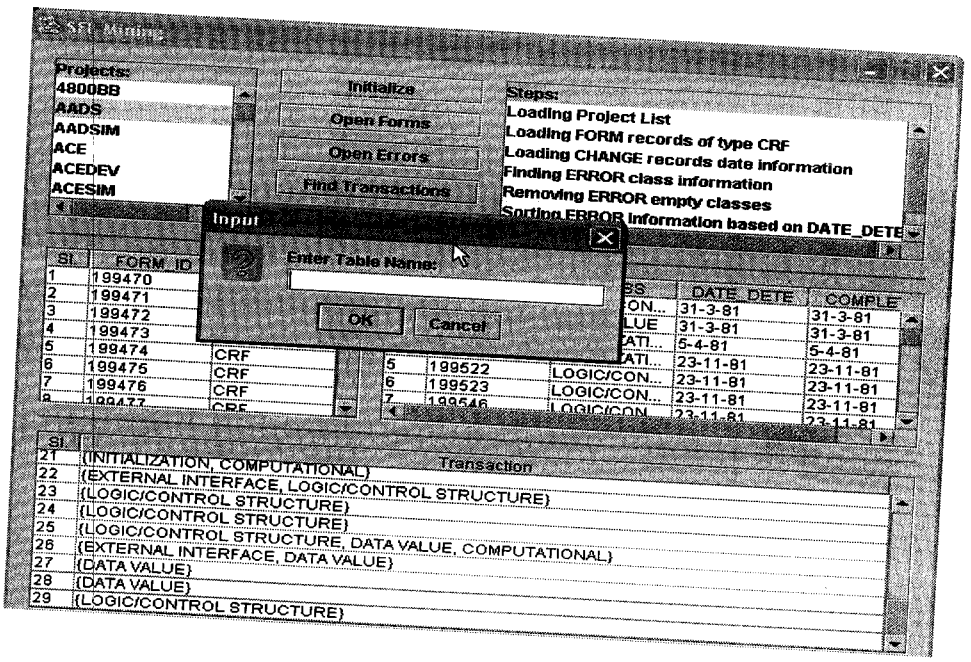


Find Transactions

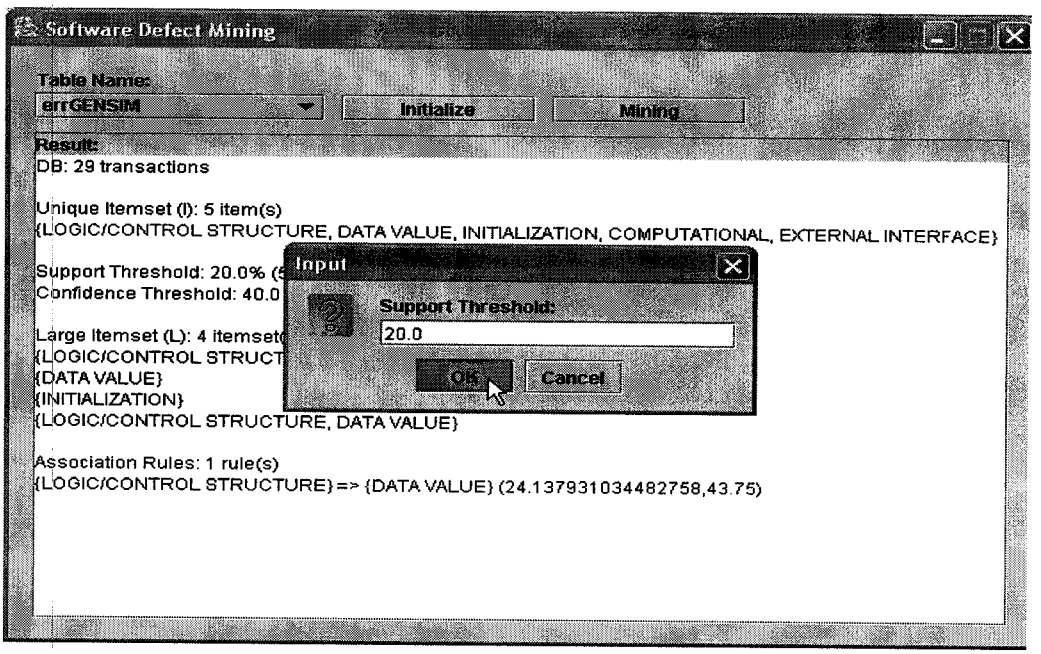


P-1529

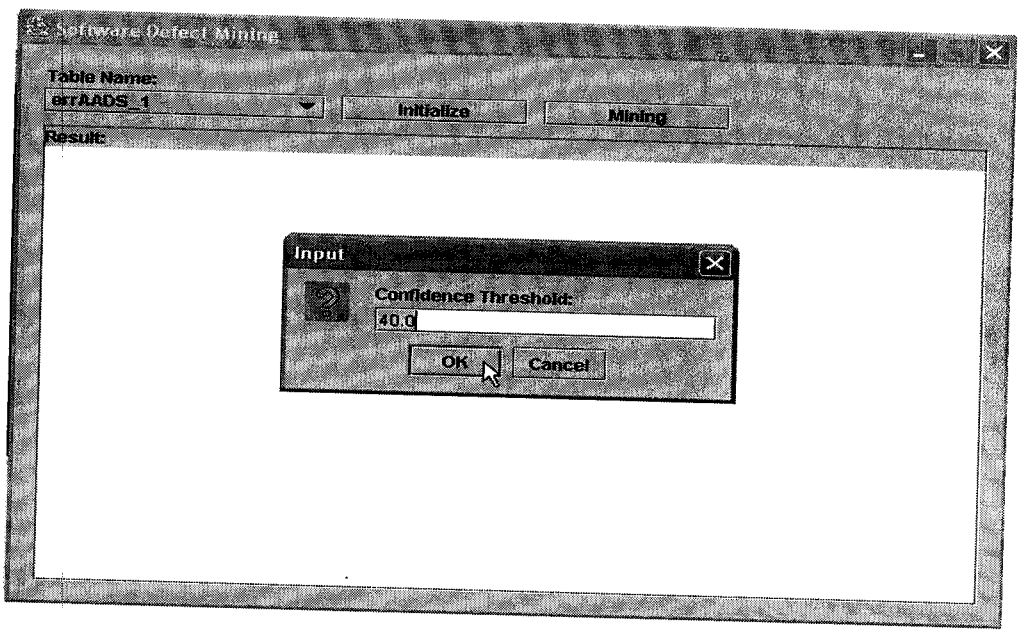
Save Transaction



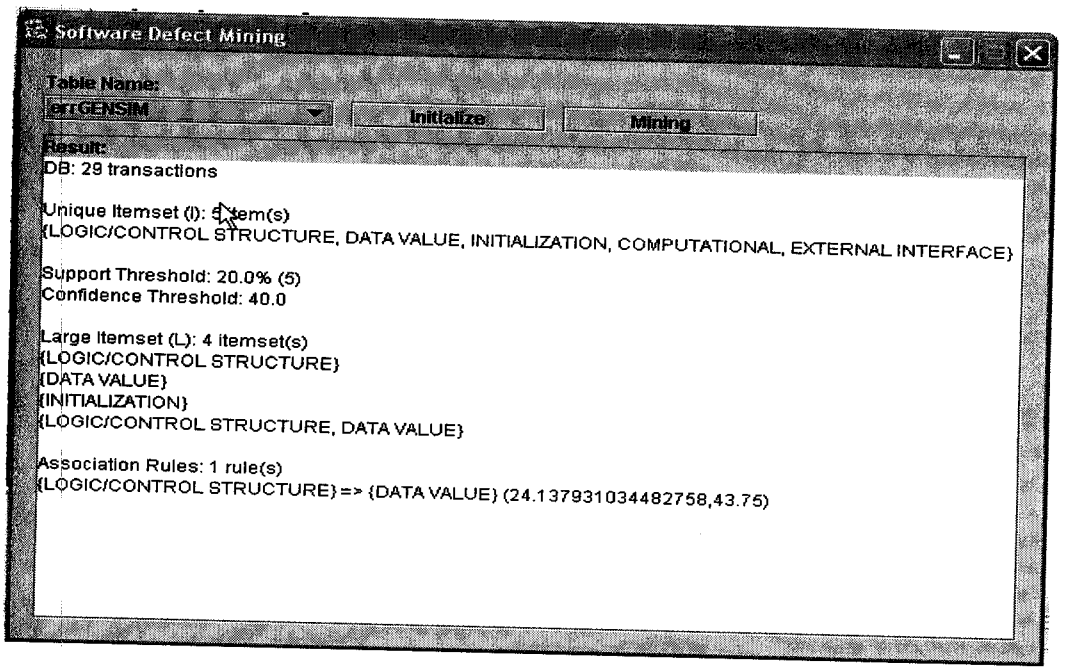
Enter Support Threshold



Enter Confidence Threshold



Association Rule Mining



Reference

REFERENCE

1. R. Agrawal, T. Imielinski, and A. Swami, "**Mining Association Rules between Sets of Items in Large Databases**", Proc. ACM SIGMOD Conf. Management of Data, May 1993.

2. K. Ali, S. Managanaris, and R. Srikant, "**Partial CAlassification Using Association Rules**", 1997.

3. Herbert Schildt, "**The Complete Reference, Java2**", Tata McGraw-Hill Publishing Company, 2002.

4. R. Srikant, Q. Vu, and R. Agrawal, "**Mining Association Rules with Item Constraints**", Proc. Third Int'l Conf. Knowledge Discovery and Data Mining, Aug. 1997.

5. C. Wohlin and P. Runeson, "**Defect Content Estimations from Review Data**", Proc. 20th Int'l Conf. Software Eng, 1998.

6. G.Heller, J. Vallet, and M.Wild, "**Data Collection Procedure for the Software Engineering Laboratory Databases**", Technical Report SEL-92-002, Software Eng. Laboratory, 1992.

7. SEL Database, http://www.cebase.org/www/frames.html/resources/SEL/

8. http://en.wikipedia.org/wiki/Java_Database_Connectivity