2011

An empirical study of software changes in open source systems

Deel M. Baral

AN EMPIRICAL STUDY OF SOFTWARE CHANGES IN OPEN SOURCE SYSTEMS

by

DEEL M. BARAL

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Software Engineering
in
The Department of Computer Science
to
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2011

In presenting this thesis in partial fulfillment of the requirements for a master's degree from The University of Alabama in Huntsville, I agree that the Library of the University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

2011-08-09

(Student Signature) Date

THESIS APPROVAL FORM

Submitted by Deel M. Baral in partial fulfillment of the requirements or the degree of Master of Science in Software Engineering and accepted on behalf of the faculty of the School of Graduate Studies by the thesis committee.

We, the undersigned members of the Graduate Faculty of the University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate on the work described in this thesis. We further certify that we have reviewed the thesis manuscript and approve it in partial fulfillment of the requirements of the degree of Master of Science in Software Engineering.

8-9-2011
Dr. Wei Li (Date) Committee Chair

8/10/11
Dr. Letha Etzkorn (Date)

8-10-11
Dr. Harry Delugach (Date)

Heggere S. Ranganath 8-10-11
Dr. Heggere S. Ranganath (Date) Department Chair

8/10/11
Dr. John D. Fix (Date) College Dean

Rhonda Kay Gaede 11/21/11
Dr. Rhonda Kay Gaede (Date) Interim Graduate Dean

ABSTRACT
The School of Graduate Studies
The University of Alabama in Huntsville

Degree: Master of Science in Software Engineering

College/Department: Science/Computer Science

Name of Candidate: Deel M. Baral

Title: An Empirical Study of Software Changes in Open Source Systems

Software changes are crucial in software systems for fixing errors, feature enhancement, improved performance, and adaption of the system to an altered environment. Software change activities are costly in terms of effort and time. Understanding software change behavior helps us develop methods to reduce the software change cost when fixing errors or altering software. In this thesis, we present the results of three software changes studies: classification, distribution and clustering. Our results show that the hierarchical classification could not perform significantly as compared to flat classification. The total changes in software systems taken followed the power-law distribution. The error changes in 80% and non-error changes in 75% of the software systems taken followed the power-law distributions. The total changes and the error changes in 80% of the systems taken followed the Weibull distribution. However, the non-error changes followed the Weibull distribution in only 40% of the systems taken. Our study discovered that frequently changed files clustered in nearly 16% of the software modules.

Abstract Approval: Committee Chair _____
Department Chair _____
Graduate Dean _____

ACKNOWLEDGMENTS

The work described in this thesis would not have been possible without the assistance of a number of people who deserve special attention. Firstly, I want to thank my parents, Dr. Kamalmani Baral and Bishnu Maya Baral, for their love and affection to bring this work towards a successful culmination. Special thanks go to my thesis advisor, Dr. Wei Li, for his proper guidance, warm co-operation and encouragement during the work. I would also like to thank Dr. Letha Etzkorn and Dr. Harry Delugach for being on my thesis committee. I would also like to thank Dr. James J. Swain for verifying the mathematical results of my thesis. I would also like to express gratitude to Dr. Paul Wolfgang from Temple University who provided me with helpful instruction to use *text tools* properly.

Finally, I would also like to thank my friends who have provided me with valuable suggestions and coordination.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

LDA- Latent Dirichlet Allocation

LSI-Latent Semantic Indexing

MALLET-MAchine Learning for LanguagE Toolkit

MaxEnt-Maximum Entropy

PHP- Hypertext Preprocessor

PLS-Partial Least Squares

REGEX-Regular Expressions

RFE -Request for Enhancement

SDN- Sun Developer Network

SVM- Support Vector Machine

TF-IDF-Term Frequency-Inverse Document Frequency

URL- Uniform Resource Locator

CVS- Concurrent Versions System

SVN- Subversions

CSV-Comma-Separated Values

XML-Extensible Markup Language

NOL-Number of Loops

NOFC- Number of Function Calls,

NOFD -Number of Function Declarations,

NOV -Number of Variable Declarations

CHAPTER 1

INTRODUCTION

Software systems are constantly subjected to changes: error fixing, feature enhancement, performance improvement, and adaption of the system to an altering environment. Software changes demand resources and account for the majority of the cost of the software [Gupta et al. 2006]. Understanding software change behavior helps us develop methods to reduce the software change cost when fixing errors or altering software. For example, we can focus testing efforts in the area of a system where there is a high concentration of changes. Furthermore, the ability to change software quickly and reliably benefits businesses [Bennett and Rajlich 2000]. Software changes can be error related and non-error related. The error changes are triggered by uncovered errors in the development, testing, or use of software systems. Non-error changes are triggered by the enhancement of features, refactoring, and other modifications of software systems which are not the result of fixing errors. In this research, we conduct experiments to understand change behavior in several open source systems.

To study the change behavior for each type of change, the changes must be classified into different categories based on the nature of the changes.

A number of studies exist on software changes from different viewpoints: distribution, automated classification, and cluster studies. Change distributions identify

the statistical distributions observed in the change data. The automated change classification process analyzes each change description using machine learning approaches [Holts et al. 2010] and puts that change description in a particular change category. Change cluster [Kothari et al. 2006] studies of software modules identify the group of modules that are prone to changes by identifying file clusters—subsets of frequently changing source files. The source lines that are changed (i.e., added or deleted) are the sources of altering the files and the modules. In this thesis, we present three studies: one on automatic change classification, one on change distribution, and one on file clustering.

Most researchers classified software changes based on the change activity associated with either fixing or not fixing errors. Kim and colleagues [Kim et al. 2008] conducted a study to automatically classify software changes as either "clean" or "buggy" on 12 open source systems. Ferzund and colleagues [Ferzund et al. 2009] classified software changes in hunks — changed regions in files, automatically as "buggy" or "bug-free" with 81 percent accuracy, based on hunk metrics. A few examples of hunk metrics are Number of Loops (NOL), Number of Function Calls (NOFC), Number of Function Declarations (NOFD), and Number of Variable Declarations (NOV). In our context, "buggy" changes are error changes and "bug-free" or clean changes are non-error changes. Li and colleagues [Li et al. 2006] performed an automated classification of 28,928 bugs (errors changes) and found a decrease in memory-related bugs, and an increase in security bugs and semantic bugs, where semantic bugs were dominant in Mozilla [2005], in comparison to the same software system more than 10 years ago.

Anderson and Runeson [2007] showed the distribution of faults (error changes) in modules that followed the power-law distribution [Clauset et al. 2009] in Eclipse data [Zimmermann et al. 2007]. Later, Zhang [2008] showed that the Weibull distribution was a better fit than the power-law distribution using the same Eclipse data.

In a file clustering study, Ying and colleagues [Ying et al. 2004] applied the association-rule mining algorithm (FP Tree) to the development history of Eclipse and Mozilla to predict the future modification of source code in a subset of source files.

In this research, we performed an automated classification of changes into many change categories based on the change descriptions in the open source data and examined the effectiveness of the four classification algorithms: SVM [Thorsten 2010], Naïve Bayes [McCallum 2002], MaxEnt [McCallum 2002], and N-Gram [Alias-i 2008]. We tested the change data for the power-law distribution [Clauset et al. 2009] and the Weibull distribution [Zhang 2008]. We investigated the file clusters of software modules by using association rules based on the H-mine data mining algorithm [Pei et al. 2007]. The goal was to find the subsets of files that were changed frequently together in the NetBeans source code. The traceability link between frequently changed subsets of source files and the source code modules (java source folders) was used to identify the changed module clusters.

1.1 Research Objectives

Our research has the following objectives:

1. Develop a software crawler tool to automatically collect change data from public websites.

Manual collection of large amounts of open source data is slow and inefficient. A crawler tool can collect information from public websites efficiently.

2. Investigate the effectiveness of automated change classification by using four classification tools at a time for all change categories.

   Classifying change data into more detailed categories will afford us the opportunity to examine changes more closely. There are several textual data classification techniques (algorithms): SVM [Thorsten 2010], Naïve Bayes [McCallum 2002], MaxEnt [McCallum 2002], and N-Gram classifiers [Alias-i 2008]. Li and colleagues [2006] performed classification of fixed runtime bugs in the Mozilla database by using the SVM, Winnow, Perceptron, and Naïve Bayes. However, their study used only one change category. In this research, we want to investigate the automated change classification of more than one change category.

3. Find the distribution of total changes, error changes, and non-error changes in the different software systems (Table 3.1 and Table 3.2) at the module level.

   The information logged on public websites has a unique bug identification number (bug ID). There are mainly two types of logged records: errors (bugs, defects) and non-errors (Request for Enhancement, task). The Request for Enhancement (RFE) includes changes related to feature enhancement, adaptive maintenance, and refactoring. We will use the term *change data* to refer to the logged changed records.

4. Identify the clusters of software modules that are prone to changes.

   The NetBeans repository consists of a collection of *changesets* in a changelog. Each changeset is a collection of changes made to one or more files in the source code repository in one commit. The changes made to the source files resulted in the changed clusters of software modules.

1.2 Research Hypotheses

Based on the research objectives, we set up the following hypotheses to test in our research:

1. The automated change classification algorithms are effective to classify changes with high accuracy (> 80%).
2. The total changes at the module level followed the power-law distribution.
3. The error changes at the module level followed the power-law distribution.
4. The non-error changes at the module level followed the power-law distribution.
5. The total changes at the module level followed the Weibull distribution.
6. The error changes at the module level followed the Weibull distribution.
7. The non-error changes at the module level followed the Weibull distribution.
8. The non-error changes were concentrated in the same modules where there was a high concentration of errors.
9. The most frequently changed file clusters tend to concentrate in a small set of modules.
10. Approximately 80% of the total changes were concentrated in about 20% of the changed modules (Pareto Principle).

The hypothesis on automated change classification answers the question: "Can automated classification be used for classifying changes with desired accuracy?" The hypotheses on the power-law distribution and the Weibull distribution help us reach the conclusion whether the same data followed one, both, or none of the two distributions. The hypothesis on cluster study provides us the evidence if frequently changing source files are located in a small set of source code modules (folder structure).

1.3 Organization of the Thesis

In the remainder of the thesis, Chapter 2 introduces concepts required to accomplish the research work and presents the literature that was related to our work. Chapter 3 discusses the open source software systems that we have examined and the change data collected from them. Chapter 4 introduces the design of the crawler tool. Chapter 5 presents the research design. Chapter 6 presents the data analysis results. Chapter 7 summarizes the thesis with the conclusion and future work.

CHAPTER 2

BACKGROUND

This chapter discusses the background theory for our research and reviews the literature that was related to our work.

2.1 Background Theory

This section elucidates the concepts that were used in this thesis. A short description of each technique and approach is provided in sections below.

2.1.1. Web Crawler

Web crawlers are software search agents that scan or crawl relevant objects of interest, such as text images and links in a web page, in an automated manner. Sample web crawlers are Yahoo! Slurp, Msnbot, Aspeek, crawler4j, and GRUB. The process of web crawling is referred to as *spidering.* Web crawlers start with a seed URL and parse the specified web page. According to Wikipedia,

"A Web crawler is a computer program that browses the World Wide Web in a methodical, automated manner."

There are two ways to implement a crawler: depth and breadth first. Crawlers are widely used by search engines in indexing web pages. SDN crawler, the tool created in our project, is a breadth-first implementation of a crawler that proceeds in a sequential manner.

2.1.2. Machine Learning

Machine learning is an approach of artificial intelligence to make the machine learn based on prior acquired knowledge. Machine Learning techniques have been used in automated text classification.

2.1.2.1. Supervised and Unsupervised Machine Learning

In Supervised Machine Learning, a training set of data is prepared and the classifier predicts the unclassified data to one of the preset categories. Supervised learning completes in two steps.

1. Train the classifier from the labeled documents.
2. Use the trained classifier to classify the unlabeled documents.

   Some examples of supervised methods are Latent Dirichlet Allocation (LDA) [Blei et al. 2003], Support Vector Machine (SVM) [Thorsten 2010], and MaxEnt [McCallum 2002].

Unsupervised learning differs from supervised learning in that it learns without the training data. Clustering is an example of unsupervised learning.

Both the supervised and unsupervised machine learning techniques require large collection of data for training, classification and clustering. A large collection of data is called *corpus*. In our study, we referred to text corpus as corpus.

2.1.3. Classifiers

The text classifiers used for the change classification in our research are as follows.

1. Support Vector Machine- SVM^{light}
2. Naïve Bayes

3. Maximum Entropy (MaxEnt)
4. N-gram

2.1.3.1. Support Vector Machine

SVM^{light} is an implementation of the Support Vector Machines (SVM) in C. SVM is a supervised learning algorithm because the classifiers classify the unlabeled documents based on the training data first. Different machine learning and classification tools may require different formats of documents. Each unique word and its frequency of occurrence in the training set corpus are considered as a *feature*. A *vector* is a set of features that describes one case in the training set corpus. In this research project, each row in the database table forms a vector and each distinct word except the stop words-words which occur most frequently in language-forms a feature.

One important feature of SVM is that it can learn independent of the dimensions of the feature space [Thorsten 2010]. A feature space is formed by a collection of features. The dimensions of the feature space are high in machine learning task, so the *Term Frequency-Inverse Document Frequency* (TF-IDF) [Thorsten 1997] is used to reduce the dimensionality of the feature space for improved performance.

2.1.3.2. Naïve Bayes

Naïve Bayes is a probabilistic model for classification tasks. This classifier uses Baye's Theorem and is based on the conditional probabilities [Qin 2006]. Triola [2009] defines conditional probability as "The conditional probability of *B* given *A* can be found by assuming that event *A* has occurred, and working under that assumption, calculating the probability that event *B* will occur."

Bayes theorem is stated as

$$P(B \mid A) = P(A \cap B)/P(A) \tag{2.1}$$

The probability of event *B* (given the probability of event *A*) can be calculated by dividing the probability of both events *A* and *B* occurring together by the probability of event *A*. This implies event *B* is dependent on the event prior to event *A*. Naive Bayes classifier works well when the dimension of feature space is high.

2.1.3.3. Maximum Entropy

Entropy refers to the state of disorder in a system. For the purpose of text classification, it refers to the amount of information in the natural text [Garc´ıa et al. 2006]. Maximum Entropy is widely used in Natural Language Processing tasks that are based on probability distribution estimation. The rationale of this classifier is to choose the uniform distribution of data which is the maximum entropy in the distribution of data. Features in the training corpus are used to set constraints for this probability distribution technique. Nigam and colleagues [1999] expressed the features functions as

$$f_{w,c'}(d,c) = \begin{cases} 0 & if \quad c \neq c' \\ \frac{N(d,w)}{N(d)} & Otherwise\,, \end{cases} \tag{2.2}$$

where $N(d,w)$ is the number of times word *w* occurs in document *d* and $N(d)$ is the number of words in *d.*

2.1.3.4. N-Gram

N-Gram refers to the subsequence of characters or consecutive characters in a word. N-Gram is similar to decomposing a longer text string into its shorter sub-text strings. N-Gram using two and three characters is bi-gram and tri-gram respectively. For example, bi-gram and tri-gram for the word **CLASSIFIER** is as below.

Bi-gram- _C, CL, LA, AS, SS, SI, IF, FI, IE, IR, R_,

Tri-gram- _CL,CLA,LAS,ASS,SSI,SIF,IFI,FIE,IER,ER_,R_ _.

Spaces (represented by _) are also considered for the construction of N-Gram. The goal of this classifier is to capture the common root of words. For example, the common root of words "computer" and "computation", when N-gram is 6, is "comput" i.e., the two words have N-grams (6-gram) in common.

2.1.4. Text Classification

Text classification involves the procedure of automatically assigning the unlabeled text to one or more of the destined categories. Text classification can also be done manually. Automated text classification is useful when a manual categorization of unlabeled text is impossible or inefficient.

According to Wajeed and Adilakshmi [Wajeed and Adilakshmi 2009], the objective of text classification is to automatically derive methods that, given a set of training documents D = {d1. . . dr} with known categories C = {c1, . . . , cq} and a new document q, which is usually called the query, will predict the query's category, that is, will associate q with one or more of the categories in C.

There are two types of classification that are commonly used in classifying text: flat and hierarchical.

In flat classification, a textual data is directly assigned to one of the categories as shown in Figure 2.1. But in hierarchical classification, first a textual data is assigned to one of the fewer classes. Next, the same textual data is assigned to the sub-categories which have subtle differences in meaning. This process continues until the textual data reaches the leaf node, as shown in Figure 2.2.

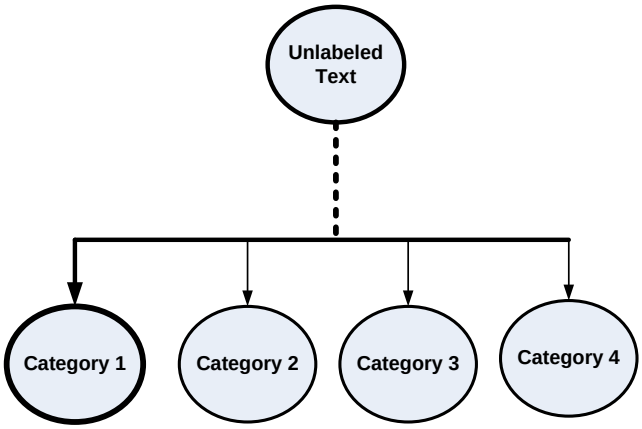

Figure 2.1: Flat Classification

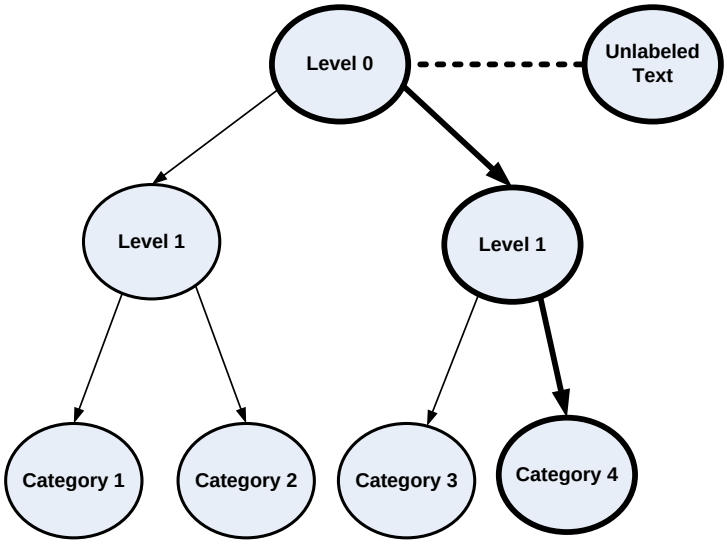

Figure 2.2: Hierarchical Classification

Note: Unlabeled text is assigned to one of the categories with bold boundary following the bold path in both classification schemes.

Based on the number of categories assigned to textual data, text classification can be single-label or multi-label. In single label classification, a natural textual data is

assigned to only one category (as shown in Figure 2.3). But in multi-label text classification, the same natural textual data can be categorized into more than one category (as shown in Figure 2.4).

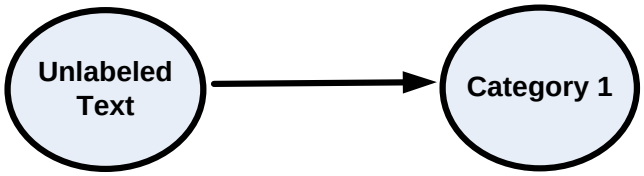

Figure 2.3: Single-Label Classification

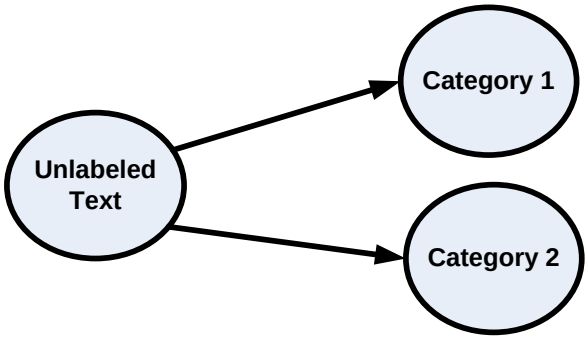

Figure 2.4: Multi-Label Classification

2.1.4.1. Training Corpus and Unlabeled Corpus

The term "corpus" is a large collection of text used in the natural language research community. This text represents data. Training corpus is the set of the training data. The sets of training data are used to build up the knowledge for machine learning algorithms. Each training set of data is assigned an answer value prior to feeding them into the machine learning algorithms. Training data is also called *labeled data* because it

has the answer values associated with it. Unlabeled data is data whose answer values are unknown and has to be determined through the use of classifiers. Collections of data whose answer values are unknown are referred to as the *unlabeled corpus*. In our study, the change category is the answer value.

2.1.5. Statistical Distribution

In this section, we discuss the statistical distribution methods used in our research.

2.1.5.1. Power-Law Distribution

"When the probability of measuring a particular value of some quantity varies inversely as a power of that value, the quantity is said to follow a power-law" [Clauset et al. 2009]. Power-law definition implies the occurrences of big events are less frequent as compared to the occurrences of small events. For example, occurrences of large magnitude earthquake are less frequent whereas occurrences of small magnitude earthquakes are more frequent. Other examples of the power law distribution include incomes of people, frequencies of words, size of craters of moon, popularity of books and music. Mathematically, if $p(x)$ is the probability distribution of a quantity x, then

$$p(x) \propto x^{-\alpha} \tag{2.3}$$

$$\text{or, } p(x) = C\, x^{-\alpha} , \tag{2.4}$$

where α is a constant and called exponent or scaling parameter is a constant parameter, C is a normalization constant. The probability distribution as described by (2.4) is called the *power-law distribution* [Clauset et al. 2009].

2.1.5.2. Weibull Distribution

The Weibull distribution is widely used in reliability engineering to determine the lifetime of products in terms of hours, cycles, miles, etc. [Weibull.com 2011]. The probability density function for a 3- parameter Weibull distribution is given below [Cousineasu 2009].

$$f(x|\gamma,\beta,\alpha) = \gamma\beta^{-\gamma}(x-\alpha)^{\gamma-1}e^{-\left(\frac{x-\alpha}{\beta}\right)^{\gamma}} \quad , \qquad (2.5)$$

where $\beta>0$ is a scale parameter, α is a scale parameter and $\gamma>0$ shape parameter.

3-parameter Weibull distribution provides better fit for any data taken as compared to one and two-parameter Weibull distribution.

2.1.6. Changelog, Changeset, and Diff File

A changelog contains the history of changes made to the source code repository. Each change made to the source code repository is called a *changeset*. A changelog [Mercurial 2011] contains the list of changesets with the following information:

1. Revision — revision number of the single repository that refers to distinct changeset.
2. Changeset ID — unique identifier for changeset in a source code repository.
3. Parent changeset — changeset of the parent working directory
4. User — Name and email address of the user who makes the change
5. Date — date and time when the change was made.
6. Summary — short description about the changeset (can contain bug ID).

Each changeset contains the following information [Mercurial 2011]:

1. Node ID — unique ID that represents the contents of a file and its position in the repository.
2. List of changed files
3. Committer — person making the changes
4. Comments — why the committer made the changes
5. Branch Name — if separate line of development

Mercurial has the *hg diff* utility to show the differences of the same file in two subsequent revisions. A *diff file* shows the difference in changes per line between two revisions (older and newer) of the same file. In a diff file, an older version of the file is preceded by "---" whereas a newer version of the file is preceded by "+++". Furthermore, the hunks-the changed region-include one or more "+" characters where an addition of lines occurred and one or more"-" characters where removal of lines occurred. An unchanged line is preceded by the white space. Each hunk area starts with the following format:

'@@ -oldFileStartLine,oldFileOffset + newFileStartLine,newFileOffset @@',

where

oldFileStartLine is the start line in the old file which is preceded by -.

newFileStartLine is the start line in the new file which is preceded by +.

oldFileOffset is the hunk size area in the old file that represents the total lines deleted in the hunk.

newFileOffset is the hunk size area in a new file that represents the total lines added in the hunk.

2.2 Literature Review

This section reviews the previous works that were related to our research. This section is divided into five subsections: crawler tools, change categories, text classification, statistical distribution studies, and cluster studies.

2.2.1. Crawler Tools

Web crawler**s** are software agents that traverse web pages and collect large amounts of information. Eichmann [1994] published the web crawler called *Spider*, as part of the Repository Based Software Engineering (RBSE) project. The project created an index of the text documents from the pool of visited URL's (Uniform Resource Locator) and employed a search functionality based on relevance. All web crawlers collect information through regular expression matching.

Web crawlers increased the network traffic heavily [Eichmann 1994]. Castillo [2004] studied the technique of crawling important pages first and other pages later to reduce network traffic. Gupta and Johari [2009] used a similar approach to collect important information first and implemented a *focused web crawler* — an agent that targets a particular topic, to collect web documents based on particular keywords.

In this thesis, a crawler tool was developed to collect the change data from the Sun Developer Network [SDN 2011] bug repositories. Our crawler had a different design from that of other crawlers in that the next link to be visited (i.e., the link to the individual bug ID), was fetched from the database (link that was previously inserted into the database). The links to the individual bug IDs were collected and stored in the database before the crawler tool fetched the individual bug ID link to collect change information. We discussed the detailed design of our crawler tool in Chapter 3.

2.2.2. Software Change Categories

Several studies have used different terms for different categorizations of software changes. IEEE [2009] used the term *defect* and provided the defect type values: Data, Interface, Logic, Syntax, Description, and Standards. Li and colleagues [2006] used the term *bug* and provided three dimensions of bug classification: Root, Impact, and Software component. Bug categories in each dimension are as follows:

- Root Causes: Memory, Concurrency, and Semantic.
- Impact: Hang Crash, Data Corruption, Performance degradation, Incorrect Functionality, and Unknown.
- Software Component: Core, GUI, and I/O.

Memory and Semantic bugs are further classified by Li and colleagues [2006] as follows:

- Memory: Memory Leak, Uninitialized Memory Read, Dangling Pointer, NULL Pointer Deference, Overflow and Double Free.
- Semantic: Missing Features, Missing cases, Corner Cases, Wrong Control Flow, Exception Handling, Processing, Typo and other Wrong Functionality Implementation.

Bezier [1990] provided a bug taxonomy and discussed bug categories as Functional, Functionality as Implemented, Structural, Data, Implementation, Documentation, Integration, System and Software Architecture, and Execution. Guo and Sampath [2008] performed web-fault classification in two large open source web applications and identified new fault categories: Data Store, Appearance, Link, Form, Compatibility, and Logic.

In our study, we combined the categories suggested by the previous studies and customized them to suit our need. We also added some new categories. The change categories that we used in our classification study are summarized in Table 4.1.

2.2.3. Text Classification

Text classification assigns text to one or more of the predefined categories. Many classifiers — tools to classify documents — can be used in text classification procedures. For instance, Support Vector Machine (SVM) [Thorsten 2010], Naïve Bayes [McCallum 2002], MaxEnt [McCallum 2002], N-Gram [Alias-i 2008], Winnow, and Perceptron are some of the commonly used automated classifiers for text classification.

The change data collected by the crawler tool were in textual format. Thus, we can apply the text classification procedure to classify change data into different categories. Several researchers have applied a text classification approach in different data. Li and colleagues [2006] classified 75,519 fixed bugs into runtime errors of two large systems: Mozilla and Apache HTTP Server, by using the SVM, Winnow, Perceptron and Naïve Bayes classifiers. The precision (fraction of documents that are correctly classified in the particular category) was 89.6%. They identified decrease in memory-related bugs, increase in security bugs and semantic bugs, and the semantic bugs being dominant in the present version of Mozilla [2005], in comparison to the Mozilla more than10 years ago.

Bo and colleagues [2009] conducted an experiment on the automatic classification of web pages using the Hidden Naive Bayes (HNB), Bayes, Support Vector Machine (SVM), K Nearest Neighbor, C4.5 and Complement class Naive Bayes. Among these six classifiers, HNB attained the best accuracy and F-measure. Hao and colleagues

[2009] showed that the Naïve Bayes algorithm, combined with the SVM algorithm, had better web text categorization in comparison to the traditional SVM. In this research, the Naïve Bayes algorithm was used for training and the SVM algorithm was used for the classification. The Naïve Bayes algorithm reduced the features in the training phase as compared to the higher dimensions generated by the SVM algorithm. Unique words and their frequency of occurrence in the training data were considered as features.

Rahmoun and Elberrichi [2007] performed the N-Gram text classification for corpuses from Reuters-21578 newswire articles and 20 newsgroups dataset. The main idea in their study was to use the multivariate chi-square distribution for the extraction of features and see the results by varying the value of "n" in n-grams of characters. The results were better than other classification procedures that involved stemming and stop words removal.

Nigam and colleagues [1999] assigned a class label (classification category), to a document using the maximum entropy technique and the results were compared with a regular and scaled Naïve Bayes classifier in three datasets: WebKB [Craven et al. 1998], Industry Sector [McCallum et al. 1998] and Newsgroups [Joachims 1997]. The comparison was based on the classification error (%) where maximum entropy had always the low classification error compared to the regular Naïve Bayes and scaled Naïve Bayes.

Cavnar [1994] reported 99.8% and 80% classification accuracy on two datasets: USENET newsgroup articles written in different languages and computer-oriented newsgroup articles using the N-gram classifier.

Pulijala and Gauch [2004] used a hierarchical text classification procedure to classify a heterogeneous collection of web contents. This paper reported an increase in the precision of 45.4% compared to the flat classification. The study was successful in assigning 70.1% of documents to their original categories using the hierarchical level-3 classifier. Sometimes, the classification procedure was more accurate with hierarchical classification than flat classification, depending on the nature of the data. Paul [2010] provided instruction to use text classification tools: SVM, MALLET, and LingPipe.

In text classification, most researchers achieved high accuracy and precision on documents such as newspapers and published articles. Based on the success of the previous studies, we decided to use the Support Vector Machine (SVM) [Thorsten 2010], Naïve Bayes [McCallum 2002], MaxEnt [McCallum 2002], and N-Gram [Alias-i 2008] classifiers for the classification of our change data [SDN 2011], to see if we could achieve similar classification accuracy. In addition, we used the hierarchical classification [Pulijala and Gauch 2004] to improve the classification accuracy further.

2.2.4. Statistical Studies

In distribution studies, two common distributions, Power-law [Clauset et al. 2009] and Weibull [Zhang 2008], were commonly used in error distribution studies.

2.2.4.1. Power-Law Distribution

Ichii and colleagues [2008] studied the in-degree and out-degree distributions in software component graphs composed of Java classes. They found the in-degree distribution followed the power-law distribution, but the out-degree did not. Chen and

colleagues showed that bug reports followed the power-law distribution [Chen et al. 2009]. In their study, the bug reports were modeled as a topological network called the Reporter Network. Then the study showed the property and degree of nodes of the reporter network that followed the power-law distribution. Clauset and colleagues [2009] presented a principled statistical framework to study the power-law behavior. They combined maximum-likelihood fitting methods with goodness-of-fit tests based on Kolmogorov-Smirnov statistics and likelihood ratios to show the effectiveness of the approach used. Later, they claimed that the commonly used methods, such as least-squares fitting, were not an accurate and reliable means to analyze the power-law data. Among 24 datasets used in their study, 7 (the same datasets used by other researchers) did not follow the power-law distribution. The mathematical result produced by their tool contradicted the conclusion reached by other researchers who claimed their respective datasets followed the power-law distribution.

2.2.4.2. Weibull Distribution

Zhang [2008] performed an experiment on the Eclipse data collected by Zimmermann and colleagues [Zimmermann et al. 2007]. He showed the distribution of software faults over software modules was better modeled by the Weibull distribution in comparison to the Pareto Principle [Pareto 1906], which was the result shown by Anderson and Runeson [2007]. The Pareto Principle is a power-law distribution [Adamic 2011]. It is widely popular in the software engineering field as an 80/20 rule.

In our study, we tested both the power-law and the Weibull distributions for the total changes, error changes and non-error changes. Furthermore, we analyzed the total changes in source code modules to test the Pareto-Principle.

2.2.5. Cluster Study

Several previous studies used the data mining techniques to mine the software change history in order to find the set of files that changed frequently in the past. Ying and colleagues [Ying et al. 2004] applied the association rule mining algorithm, FP Tree, to determine the set of files that frequently changed together in the development history of Eclipse [Eclipse 2011] and Mozilla [Mozilla 2005]. Then, they used these sets to predict the pertinent source code for upcoming modification tasks. Zimmermann and colleagues [Zimmermann et al. 2004] developed the ROSE tool that used the Apriori Algorithm [Agrawal and Srikant 1994] to compute the association rules that correctly predicted which files would be changed next (with 26% precision) and functions or variables to be changed next (with 15% recall).

In our study, we used the H-Mine algorithm [Pei et al. 2007] to find the subsets of files that were changed frequently together in the past and used these subsets to identify the cluster of modules that were the most prone to changes.

2.3 Our Approach

Our research was related to the work described in Section 2.2. Our work was based on Li and colleagues [Li et al. 2006], IEEE standard [IEEE 2009], Paul [Paul 2010], and Clauset and colleagues [Clauset et al. 2009], but our study was different from the previous work in several ways. First, the software change categories in Table 4.1 were more generalized and covered a wider range of non-error change categories such as the RFE, refactoring, and adaptive categories. Second, most previous work examined a few hundred bugs by sampling and then employed automated classification on a large number

of bugs. In our study, all 3761 bug IDs for the two open source systems (Java Server Faces and package classes_lang of Java SE JDK/JRE) were classified manually to use as a benchmark for comparison with the classification results from the automated procedure. In addition, we experimented with category removal and the supervised selection of training corpus approach to see whether the hierarchical classification accuracy could be increased above 80%. Third, Zhang [2008] and Anderson and Runeson [2007] performed an experiment on the Eclipse data to see the distribution of software faults over modules. Our research was aimed at more datasets from many open source systems: Sun Developer Network systems and NetBeans. Lastly, we investigated the clustering of software modules that were frequently affected by the error changes and non-error changes. Our approach involved the use of the H-Mine algorithm [Pei et al. 2007] to find the subsets of files that changed frequently and used the subsets of files to identify the modules that were most prone to changes.

CHAPTER 3

THE OPEN SOURCE SYSTEMS EXAMINED AND DATA COLLECTION

In this chapter, we present the open source systems examined and the change data that we collected from them. This chapter is divided into the following sections:

Section 1: Software repositories and software systems

Section 2: The crawler tool design

Section 3: The data samples

3.1 Software Repositories and Software Systems

Software changes are available in software repositories. Software repositories encompass source control repositories and bug repositories, along with any trail of archived information in software development and maintenance. The historical information recorded in software repositories contains valuable information about the evolution of software projects that supports software research and practices [Hassan 2008]. Any Source control repositories such as the Concurrent Versions System (CVS), Subversions (SVN), and Mercurial repositories include changes to the source code during the development and evolution of software projects. Bug repositories contain the history of reported bugs, feature enhancement requests and resolution history of bugs

[Hassan 2008]. The Sun Developer Network (SDN) [SDN 2010], Bugzilla [Bugzilla 2011] and Jira [Jira 2011] are some examples of bug repositories.

Most open source software systems, such as Eclipse [Eclipse 2011], provide easy access to their software repositories, but some provide only limited access to their repositories. For example, the SDN provides access to bug repositories only. Access to some open source code repositories at Java.net [Java.net 2010] requires permission from the project manager.

In our research, we studied the bug repositories of SDN [SDN 2010] and NetBeans [NetBeans 2011]. SDN only provides access to bug repositories, but NetBeans provides easy access to its bug repositories, source code repositories (Mercurial), and changelog, which contains the history of changes made to the source code. Change data of the SDN systems and NetBeans are published in http://bugs.sun.com/ and http://netbeans.org/bugzilla/query.cgi, respectively. NetBeans change data is available for download in Comma-Separated Values (CSV), and Extensible Markup Language (XML) file formats. We obtained the NetBeans change data in the CSV format from its bug repository. SDN publishes its change data only on the website. Thus, we had to develop a crawler tool to automatically collect the change data from the SDN website. The design of the tool is described in Section 3.2.

Table 3.1 shows the 65 system names from which the SDN crawler tool collected the change data. Table 3.2 shows the NetBeans system descriptive statistics. The NetBeans data was used for the cluster analysis for testing the hypotheses related to the subsets of frequently changed files. Both tables present the error changes in the *Error* column, non-error changes in the *Non-error* column and total change counts for each

system in the *Total* column. The descriptive statistics of error and non-error at the module level (folder structure) for each system listed in Table 3.1 is presented in Appendix A.

Table 3.1: Statistics of SUN Software Systems

Product Name	Error	Non-error	Total
Connected Limited Device Configuration	1022	34	1056
Custom Doclets For The Javadoc Tool	344	71	415
Gcc For Sparc Systems	13	0	13
Hotspot	14601	1946	16547
IDL Compiler	78	12	90
Inactive Categories	23	1	24
Java Access Bridge For Assistive Technology	146	21	167
Java Advanced Imaging)	1610	256	1866
Java Api For Xml-Based Rpc	677	29	706
Java Architecture For Xml Binding	766	99	865
Java Authentication And Authorization Service	114	27	141
Java Business Integration (JBI)	443	23	466
Java Cryptography Extension	521	87	608
Java Data Objects	105	7	112
Java Database Connectivity	1172	172	1344
Java Deployment	1565	306	1871
Java EE (Enterprise Edition)	3249	212	3461
Java Generic Security Services	223	59	282
Java Guides	3052	217	3269
Java IDL	1416	112	1528
Java Management Extensions	697	200	897
Java Management Extensions Remote API	57	7	64
Java ME Fragmentation	2	4	6
Java ME SDK	1560	174	1734
Java ME TCK Framework	36	16	52
Java Media Framework	2015	106	2121
Java Message Queue	4535	497	5032
Java Naming And Directory Interface	458	55	513
Java Plugin	5197	442	5639
Java SE (JDK/JRE)	72968	11509	84477
Java SE Timezone Update Tool	67	54	121
Java SE Visualvm	10	2	12
Java Secure Socket Extension	708	96	804

Table 3.1 (cont'd…)

Java Servlet	423	80	503
Java Shared Data Toolkit	3	2	5
Java Tools Documents	54	4	58
Java Tutorial	25	7	32
Java Web Server	2118	172	2290
Java Web Services Developer Pack	1152	42	1194
Java Web Start	2343	381	2724
Java3D	2938	74	3012
Javabeans Activation Framework	88	23	111
Javahelp	666	113	779
Javamail	301	128	429
Javaserver Faces Technology	392	34	426
Javaserver Pages Standard Tag Library	20	3	23
Jax-WS	998	68	1066
JAXP	1644	156	1800
JAXR	327	17	344
Jini Network Technology	1560	545	2105
Jiro	224	33	257
JT Harness (JavaTest)	205	75	280
K Virtual Machine	688	68	756
Mobile Information Device Profile	5350	510	5860
Open ESB	713	77	790
Openinstaller	883	222	1105
Personal And Embedded Java	1026	107	1133
Portal Server	8120	1095	9215
Scripting	57	11	68
Service Registry	901	67	968
Soap With Attachments Api For Java	178	17	195
Standard Javadoc Doclet	1062	387	1449
Sun Java Studio	47966	5095	53061
Sun Java System Portalserver	14216	1975	16191
Sun Studio	1230	278	1508

Table 3.2: Statistics of NetBeans

Product Name	Error	Non-error	Total
NetBeans	164375	30475	194850

3.2 The Crawler Tool Design

The crawler tool "SDN Crawler" developed in this research work was used to collect the change data from the SUN bug repositories [SDN 2011]. It is coined after the name of the repository [SDN 2010] from where it collects the change data.

The data flow diagram of the SDN Crawler is illustrated in Figure 3.1. The tool starts with the seed URL (i.e., the starting URL), gets the content of that URL, processes the raw data and inserts the information into the database that we created to store the collected data. The tool works in an iterative fashion, such that the information is gathered first in D1 with the use of the seed URL; using D1 to gather information in D2 and finally gather the information in D3 by using the information available in D2.

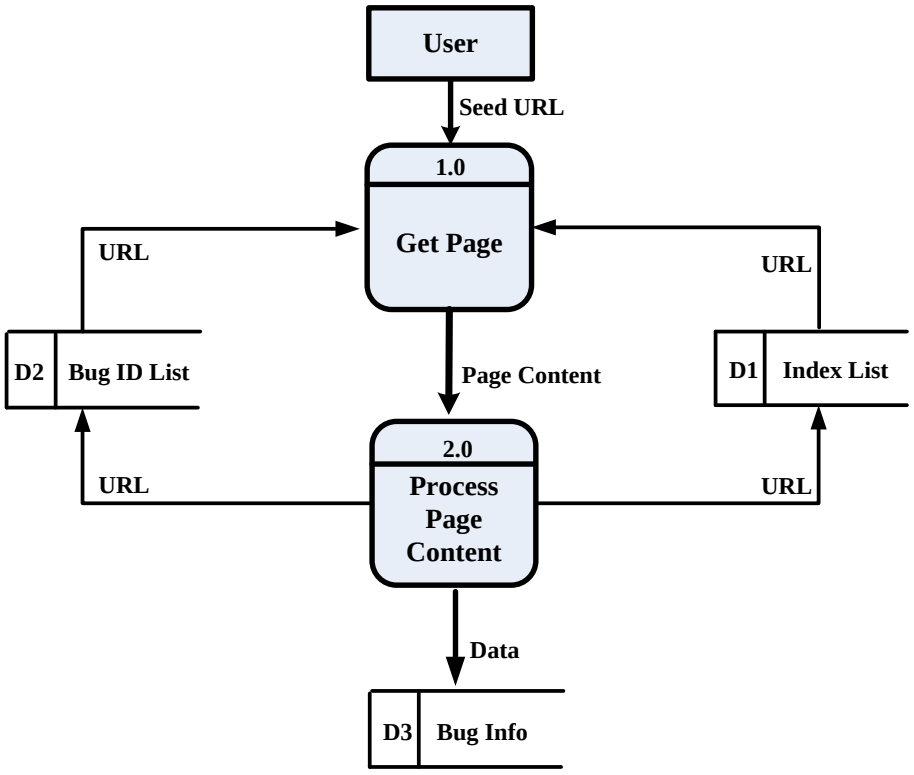

Figure 3.1: Data Flow Diagram of SDN Crawler

The algorithm for the SDN crawler is as below.

1. Provide the seed URL or the root URL, where the crawler can start.
2. Collect the entire index list available from the seed URL web page.
3. For each index list, parse all the bug ID available.
4. For each logged data
    a. Parse out the Synopsis field(i.e., short summary)
    b. Parse out the Category field (i.e., the product/category name)
    c. Parse out the State field (i.e., the status)
    d. Parse out the Priority field(i.e., severity)
    e. Parse out the Description field (i.e., explanation)
    f. Parse out the Evaluation field (i.e., developers comment)

3.2.1. Working Steps of the SDN Crawler

The SDN crawler tool collected the change data in the four steps.

The first step was the collection of all index lists. Each index list contained a collection of bug IDs. The base URL that contained of 124 index lists as Index_1, Index_2, Index_3, …, Index_124 was "http://bugs.sun.com/bugdatabase/bugIndexes.jsp?start= ".

The second step was the collection of all bug IDs in each index list. Each index list URL contained the collection of bug IDs. For example the URL "http://bugs.sun.com/bugdatabase/bugIndexes.jsp?start=1" contained the bug IDs 1169723, 1183103, 1183232, …, and 2017346.

The third step was the formation of the individual URL for each bug ID, which was stored in the database. For example, the specific URL for the bug ID 1169723 was "http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=1169723."

The final step involved the extraction of the change information **(**the bug ID, Synopsis, Category, State, Priority, Description, and Evaluation) from the web page content of each individual URL obtained in step 3. The category field contained the product name and the module (the folder structure where the change data were grouped together).

3.2.2. Challenges in the Development of the SDN Crawler

We encountered many challenges in designing and developing the SDN crawler. The following are samples of these challenges:

1. Denial of service attack in server while crawler tool was running.
2. Web site responded slowly.
3. Large volume of logged information in bug repository made it difficult to collect them.
4. Design of website changed during the work on progress. Initially, Sun's website had the cap on the display of bugs to 1000 bugs in each category. But now, there is no limitation in obtaining the logged information.

When the SUN website was launched after its redesign, the three problems: the denial of service attack, slow website response, and a cap on the display of bugs went away. We collected the raw data, i.e., the data embedded in the html syntax, in incremental steps from the SUN website into the local database, and then processed the

raw data into an useable format. This allowed us to decrease the time our crawler had to stay in touch with the SUN web server.

3.2.3. Methods to Obtain the Change Data

Change data is published in web pages. Pattern matching is a powerful technique to match character strings. For example, the regular expression "[0-9]+" matches any length of digits.

Table 3.3 shows an example of one change datum that was collected. Our crawler tool collected the bug ID, synopsis, category, state, priority, and description and evaluation fields for each logged change from the SUN bug repository. We used the term *Bug Report* to collectively represent all these fields.

Table 3.3: A Sample Bug Report

bug ID	4599651
Synopsis	getRenderNamesForComponents returns class names
Category	jsfaces:jsfaces_ri
State	11-Closed, Unverified, bug
Priority	3-Medium
Description	RenderKit.getRenderNamesForComponent returns classnames when it should return renderer names (which in turn refer to class names).
Evaluation	use "componentype" string only. (not concatenated with "renderer");

```
<font face="">4599651</font>
</td>
<td align="right" class="grey5">
<font face=""><b>Votes</b></font>
</td>
<td align="left">
<font face="" color="#ff0000">0</font>
</td>
</tr>
<tr>
<td align="right" class="grey5">
<font face=""><b>Synopsis</b></font>
</td>
<td align="left">
<font face="" >getRenderNamesForComponents returns class names</font>
</td>
</tr>
```

Figure 3.2: Sample HTML Page Content

The required change data is embedded with the html tags in webpage as shown in Figure 3.2. The SDN Crawler tool extracts the information embedded inside the HTML tags through regular expression matching.

The choice of language for this research is PHP. PHP is convenient when it comes to processing pattern-matching using regular expressions.

3.2.4. Graphical User Interface

This section illustrates the user interface provided to the user with the necessary instructions to obtain the complete processed change information. The SDN Crawler starts with the *index.jsp* page. The tool completes its task in several steps that are listed below.

STEPS

1. Create the data source name-The data source name is required in order to create the database tables.

2. Create Tables-Necessary tables are created: *indexlist*, *bug_list_table*, *bug_detail*, and *bug_process*.
3. Collect change index list- This step inserts the url of each index list to the table *indexlist*. Each row of *indexlist* table contains a url of the web page that contains the link to each bugID url. Each index list contains around 4000 bug IDs, except for a few lists at the end.
4. Collect all bug IDs -This step inserts the entire bug IDs of the SDN products into the *bug_list_table* that are available in a bug website.
5. Collect all change information- This step inserts the raw information that was directly obtained with the regular expression. The desired information hides around the html syntax. For example, the tool collects the state and synopsis of the logged change from the following format.

   State:

   ```
   </b></font>
       </td>
       <td align="left">
          <font face="">3-Medium
   Synopsis:

    <title>Bug ID: 1169723 Cursor position vs horizontal scrolling</title>
   ```

6. Process change information- The processing of raw information in step 5 could have been done before inserting it into the database. This technique was discarded because collecting the information online and processing each piece of change information would drastically increase the time that the tool takes to complete its task. Thus, raw information was inserted to the database in step 5

and that information was processed in step 6. The state and synopsis of the above bug id 1169723 after processing is described below.

bug_id: 1169723

State: 3-Medium

Synopsis- position vs horizontal scrolling

The change information obtained in step 6 is then used for the classification.

3.3 Data Samples

In this sub-section, we present the change data that the SDN crawler tool collects. The change data is processed and stored in a database for further use in our study. Some examples of change data samples are presented below.

Data Sample: 1

bug ID	1184392
Product	Java SE (JDK/JRE)
Package	classes_java
Priority	1-Very High
State	Closed Unverified bug
Synopsis	Error reading a particular .gif image
Description	The following url contains an image that webrunner can read: http://www. customer .uiuc.edu/General/NCSAHome.html you get the message: Error oak.lang.NullPointerException reading http://www. customer .uiuc.edu/General/Icons/SearchIcon.gif
Evaluation	X bitmap image with problems & syntax error in tar reference
Bug url	http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=1184392

Data Sample: 2

bug ID	2037215
Product	Hotspot
Package	runtime_system
Priority	3-Medium
State	Closed Verified bug
Synopsis	Error loading lib and Problematic Thread error then it hanged. Intermittent Fail
Description	See Description Note of parent CR 4373354.
Evaluation	See Evaluation note of parent CR #4373354
Bug url	http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=2037215

Data Sample: 3

BugID	4160858
Product	Jini Network Technology
Package	net_jini_discovery
Priority	4-Low
State	Closed Verified request for enhancement
Synopsis	LookupDiscovery socket manipulation enhancements
Description	LookupDiscovery should cause setSoTimeout to be called on the sockets it opens directly for unicast discovery.It would also be better if the ResponseListener thread didn't attempt to perform discovery directly, but instead queued the socket for the UnicastDiscoverer to process. That way there's less chance of overflowing the OS accept queue.
Evaluation	Yep.
Bug url	http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4160858

CHAPTER 4

RESEARCH DESIGN

In this chapter, we discuss how the research is designed for the three tasks: software change classification, test of the change distribution, and finding the clusters of software modules that are most prone to changes.

One of our research goals is to classify change information into various software change categories so that we can investigate the change distribution in each category. Manual classification is accurate, but inefficient. We set up an experiment to automatically classify change information using flat and hierarchical classification methods with the help of the four different classifiers: SVM [Thorsten 2010], Naïve Bayes [McCallum 2002], MaxEnt [McCallum 2002] and N-Gram classifiers [Alias-i 2008] as described in Section 2.1.3. To verify the automated classification accuracy, we manually classified 3761 logged changes from the Java Server Faces Technology and the package classes_lang of Java SE (JDK/JRE) [SDN 2011]. The manual classification results were compared with that of the automated classification to gauge the accuracy of the automated classification.

4.1 Software Change Categories

Several studies have used different terms for different categorizations of software changes. The categorization of changes varies upon the type of software being developed. Li and colleagues [2006] used the term *bugs* to represent error changes. These researchers proposed three dimensions of bugs and bug categories, which were discussed in Section 2.2.2. IEEE [2009] and used the term *defects* to represent the changes. It published the defect types as Interface, Logic, Description, Syntax, Standards, and Others. In our study, we combined these two methods and added a few categories of our own. We have included the Interface, Standards and Configuration categories from [IEEE 2009] and removed the Description, Syntax, and Logic categories because of the nature of the changes encountered in our study. We also removed the Corner Cases, Typo and Processing categories from [Li et al. 2006]. Li and colleagues used six categories to cover memory errors: Memory Leak, Uninitialized Memory Read, Dangling Pointer, NULL Pointer Deference, Overflow and Double Free. We grouped them into one Memory category because of the nature of errors in our data. The Missing Features, Missing cases, Wrong Control Flow and other Wrong Functionality Implementation of Semantic categories [Li et al. 2006] were grouped into the Functional Category in our study. We added the Exception Handling and Conceptual categories. Table 4.1 summarizes the categories that we used in our study.

Pulijala and Gauch [2004] reported an increase in bug classification accuracy when hierarchical classification was used. Thus, we built a hierarchal structure of changes to experiment with the hierarchical classification. Our hierarchy is shown in Figure 4.1. We grouped the Conceptual, Functional and Exceptional categories into the

Semantic category because they were all related to failures to meet the design requirements. The Enhancement, Adaptive, and Refactoring categories were grouped into the Maintenance category, because they were all non-error changes.

Table 4.1: Software Change Categories

Change Category	Description
Enhancement	Features required to be added to the software.
Adaptive	Changes required for the system to adapt in altered platforms.
Refactoring	Software altered to improve its internal behavior such as to remove unnecessary, inefficient algorithms and poorly constructed code [Pressman 2010].
Test	Bugs encountered in writing tests cases and during test such as regression test.
Documentation	Incorrect, incomplete, misleading information and typographical mistake in specification and java documentation.
Internationalization	Bugs related to adaption of software for various languages and locale specific components. Resource bundles contain locale-specific objects.
Memory	Bugs related to memory: buffer overflow, access violations, usage of an uninitialized variable, memory leak, dangling pointer, null pointer deference, double Free [Li et al. 2006].
Standards	Failure to meet the standards with the software version, coding convention and failures due to other vendors [IEEE 2009].
Interface	Bugs in the implementation of interface (between sub-routines, software modules, internal and external software components, software and Operating System etc). For example, incorrect value returned during function call, incorrect or insufficient parameters passed, and incomplete or incorrect message sent or displayed [IEEE 2009].
Concurrency	Bugs that occur in multithreading or multi-process environment, including data race, deadlock and synchronizations [Li et al. 2006].
Configuration	Bugs inserted during product build or packaging [IEEE 2009].
Conceptual	General wrong concept employed in coding. Any bugs due to understanding of programming language concept, incorrect logic employed such as infinite loops and recursion, typo other than in documentation category.
Functional	Features not required, not implemented and not according to specification [Li et al. 2006].
Exception	Do not have proper exception handling; program cannot progress because of exception or expected exception [Li et al. 2006].
Miscellaneous	Any other reasons mentioned than above.

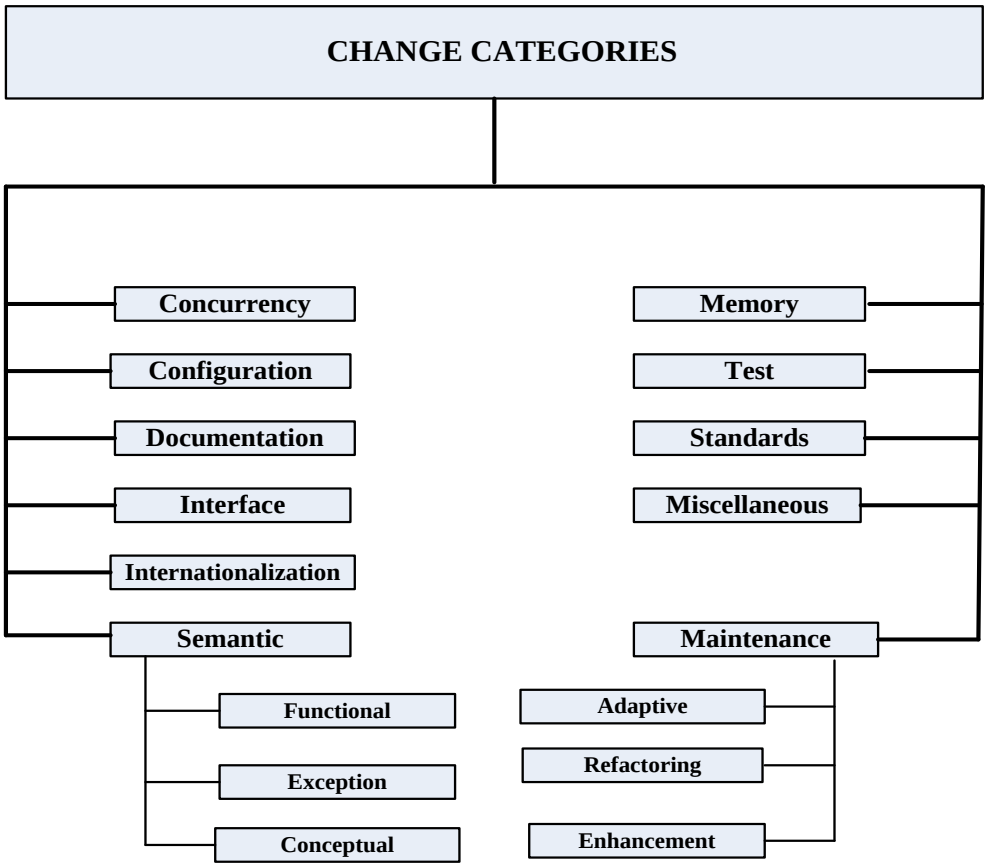

Figure 4.1: The Hierarchy Structure of Change Categories

4.2 Change Classification Study

A bug report, as illustrated in Table 3.3, was assigned to one category in Table 4.1 based on the *synopsis* field in the report. Synopsis is the description summary in a bug report. The *description* field contains detailed information, such as how to reproduce the bug, test cases, source code, and the platforms where bug was found. The *evaluation* field contains the developer's comments regarding the bugs or changes made. Occasionally, issues relating to faulty design, unclear specification and documentation were pushed into the repository by vendors. The cause for software changes are

identified by developers in the evaluation field. The description and evaluation fields have high significance in the manual change classification because they provided the information about the reported issues in detail. However, in the automated change classification, these two fields created high dimension features for the text categorization tool because of a large amount of text in them. High-dimensional feature spaces reduce the classification accuracy of text classification tools [Slonim and Tishby 2001]. Thus, only a synopsis field was used in the automated classification study.

Theoretically, each change description can be classified into more than one change category because its characteristics may map to multiple change categories. This approach of classifying a change into multiple change categories is known as the *multi-label classification* [Wajeed and Adilakshmi 2009]. The *multi-label* classification approach did not fit our need, so we did not use it.

Our change categories were given priority numbers because one change category could be the source to another change category. For example, a problem in the documentation, e.g., requirements specification, resulted in the wrong implementation of a feature. The wrong implementation of a feature was categorized as a Semantic category (refer to Table 4.1). Thus, a Documentation category had a higher priority than a Semantic category because an error in documentation led to the wrong implementation of a feature. The change priorities are

Priority 1 (Enhancement): The state of the logged change, bug report as in Table 3.3 may have Request for Enhancement. The developers assign state to logged changes and become the strong evidence for us to classify the change description to Enhancement change category.

Priority 2 (Documentation): Changes in documentation and specification act as a new source to introduce errors such as the wrong implementation of a feature.

Priority 3 (Adaptive and refactoring): Adaptive and refactoring can introduce errors in software such as semantic, interface, data, and memory.

Priority 4 (Remaining Categories): Internationalization, Memory, Standards, Interface, Concurrency, Data, Configuration, Test, Semantic (Conceptual, Exceptional and Functional) and Miscellaneous are assigned the same priority.

Change priorities remove the problem that a bug report might be assigned to multiple change categories because the description field in the bug report may fit to multiple change categories. Thus, one bug report was assigned to only one change category. This approach is known as the *single-label classification* [Wajeed and Adilakshmi 2009]. Change priorities were used in both the manual and automated classifications to guarantee that one bug report was assigned to only one change category.

4.2.1. Change Categories in the Manual and Automated Classification

Manual classification utilized all the change categories in Table 4.1, taking the change priorities into account, to classify each bug report to one change category. The automated classification utilized reduced the number of change categories compared to manual classification. Rahmoun and Elberrichi [2007] showed improvements in classification accuracy by removing the categories that were close together, i.e., small separation of concern. We removed some change categories for the automated classification. In Section 4.2, we presented the change priorities. The classifiers that we used did not have the ability to assign priority to our change categories. Thus, we removed change categories with Priorities 1, 2, and 3 and considered only the Priority 4

change categories in the automated classification. In addition, we removed the Test and Miscellaneous categories in Priority 4 for the automated classification. Test bugs were usually introduced into the test infrastructure but not in the actual source code of the targeted system. The Miscellaneous category contained the heterogeneous characteristics of changes so that the classifier might not be able to classify a miscellaneous category correctly. Thus, the Miscellaneous category was also removed from the automated classification. The final set of change categories involved in the automated classification were Concurrency, Configuration, Interface, Internationalization, Functional, Exception, Conceptual, Memory, and Standards.

For gauging the automated classification accuracy, we excluded those bug reports used in the manual classification that were assigned to the change categories not in automated classification. The classification accuracy of the automated classification is the ability of the classifiers to correctly assign change categories to the remaining bug reports that already received change category from the manual classification.

4.2.2. Manual Classification

Although, classifying changes manually can achieve high accuracy, it is not an efficient method for classifying large number of changes because it requires a greater amount of time of trained personnel. We used the synopsis, description, and evaluation fields in the bug report to manually classify all 3761 bugs from the two selected systems: Java Server Faces Technology and the package classes_lang from Java SE (JDK/JRE) [SDN 2011], to different change categories while taking change priorities into consideration. The manual classification results formed the basis to gauge the accuracy

of the automatic classification. Two sample bug reports are shown below to explain in detail how the manual change classification was performed.

Bug Report: 1

bug ID	4619152
Product	Java Server Faces Technology(jsfaces)
Package	jsfaces_ri
Priority	3-Medium
State	Fix Delivered request for enhancement
Synopsis	generate ID if not present in markup.
Description	* id is not a required tag attribute. If not present, generate a unique id and store it as the id property on the component.
Evaluation	make "id tag attribute not required for component tags
Bug url	http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4619152

In this bug report (bug ID: 4619152), the synopsis showed the addition of a feature to the Java Server Faces Technology product. The state of the logged change, Request for Enhancement (RFE), was strong evidence to categorize it into the Enhancement category in the flat categorization. In the hierarchical categorization, it fell into the Maintenance category.

Bug Report: 2

bug ID	4243974
Product	Java SE (JDK/JRE)
Package	classes_lang
Priority	4-Low
State	Closed Will Not Fix bug
Synopsis	(1.1) java.lang.Math.acos() does not conform to java specs
Description	According to "Java Language Specification" for Math.acos() "If argument is NaN or its absolute value is greater than 1, then the result is NaN."However, actual results are different: Math.acos(-10) = 0 Math.acos(10) = 0
Evaluation	This is 1.1.X only. xxxxx@xxxxx 1999-06-24
Bug url	http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4243974

The synopsis in this bug report (bug ID: 4243974) showed that an implemented feature did not conform to the Java specifications. The feature was wrongly implemented and did not meet the design requirements. Therefore, it was categorized into the Functional category (see Table 4.1 for description) in the flat classification and to the Semantic category of the hierarchical classification.

4.2.3. Automated Classification

Classifying changes to various categories afford us the opportunity to study software changes in finer detail than otherwise possible. Ideally, we want to classify a large quantity of changes using automated tools. In this section, we present our study of automated change classification.

4.2.3.1. Text Classification Tools

For the automated classification, we used four classifiers: SVM [Thorsten 2010], Naïve Bayes [McCallum 2002], MaxEnt [McCallum 2002], and N-Gram classifiers [Alias-i 2008]. These four classifiers came as a bundled package named *text tool*s [Paul 2010]. The *text tools* are supplemented with the documentation of the classifiers that were included in the package.

4.2.3.2. Methods of Automated Classification

Two methods were used in the automated classification: *flat* and *hierarchical*. In the flat classification, a change description was directly assigned to one of the ten categories: Configuration, Concurrency, Data, Internalization, Interface, Standards, Memory, Conceptual, Exception, and Functional. These categories came from Table 4.1. In the hierarchical classification, a change description was assigned to any of the

categories mentioned in the flat classification except Conceptual, Exception and Functional categories, which were first classified into another level of the hierarchy called the Semantic category. A change description that fell into the Semantic category was further classified into one of three categories: Conceptual, Exception and Functional. Thus, on the first attempt of the classification, which we called the *level-1 classification;* change descriptions were classified into the Data Configuration, Concurrency, Internalization, Interface, Standards, Memory, and Semantic categories. On the second attempt, which we called *level-2 classification*, only the semantic categories were further classified to Conceptual, Functional, and Exceptional categories. In the software systems we studied, there were a large number changes related to Semantic category. The purpose of the hierarchical classification was to reduce the misclassification rate within the semantic categories, which should improve the overall accuracy of the automated classification.

4.2.3.3. Iteration Plan

We planned to have two training cycles to observe the classification accuracies of different classifiers. Each training cycle used a new set of training data (corpus) so that the tool learned something new each time. We used two methods to prepare the training corpus (the training data set). In the first method, a corpus was selected manually based on a group of words that represented the change category. The advantage of this approach was that all the change categories had their presence in the corpus. We used the term *supervised selection* to represent the training corpus prepared manually. In the second method, the training corpus was selected randomly from the database table. The random selection of rows in the database table was done by the SQL *RAND ()* function.

The same number of training data items were selected in both iterations. We used the term *random selection* to represent the training corpus prepared through the use of the SQL RAND () function.

4.2.3.4. Formation of Training Corpus and Unlabeled Corpus

The collected bug reports from all 65 software systems in Table 3.1 were stored in one database table called *bug_processed.* A row in a table was a bug report. Two fields of each row: bug ID and synopsis, were used by the tool for the classification.

In the manual classification, we created separate tables for the two systems: Java Server Faces Technology and the package classes_lang of Java SE (JDK/JRE), and assigned a change category to each bug report in the two systems. We used supervised and random selection procedures to select the labeled bug reports (bug reports whose categories were already determined from manual classification), to prepare the training corpus. Table 4.2 shows the sample training corpus used in the manual classification. Each synopsis field had an associated value in the *Change_Category* field, indicating the labeled documents. We created the training corpus in a suitable format required by the *text tools* Paul [2010].

Table 4.2: Training Corpus

Bug ID	Synopsis	Change_Category
4410846	Unstarted Thread causes memory leak	Memory
4212219	stopped thread does not release synchronization lock	Concurrency
4243950	(1.1) java.lang.Math.log() in violation of spec	Semantic

An unlabeled corpus is a collection of unlabeled bug reports (bug reports whose categories are to be determined). For the preparation of the unlabeled corpus, the training

set of data were isolated from the database tables in the manual classification, and the *Change_Category* field was left empty for remaining records. Table 4.3 shows a snapshot of the unlabelled corpus. The goal of the automated classification was to classify the bug IDs to the same categories that the manual classification did.

Table 4.3: Unlabelled Corpus

Bug ID	Synopsis
4219639	Spelling mistake in Exception
4216125	getResource of an URL with ".." or "."
4154676	Incorrectly rounds huge float or double literal

The same set of labeled corpuses and unlabeled corpuses were used in the flat and hierarchical classification to compare their classification accuracies.

4.2.3.5. Procedure to Calculate the Accuracy of the Classifiers

Lewis [1991] provided the methodology to calculate the performance measures of the classifiers in text categorization and Yang [1991] and evaluated precision, recall, fallout, error, and accuracy for the evaluation of text categorization in binary classifiers. These performance measures are calculated from the contingency Table 4.4 [Lewis 1991].

Table 4.4: Contingency Table

		Expert Decision	
		Yes	No
Classifier Decision	Yes	x	y
	No	y’	x’

x = number of documents correctly classified to the particular category.

y = number of documents incorrectly classified by the classifier from the category.

y'= number of documents incorrectly rejected by the classifier from the category.

x'= number of documents correctly rejected by the classifier from the category.

Precision measures the fraction of documents that are correctly classified in the particular category.

$$\text{Precision} = x / (x + y). \quad (4.1)$$

Recall measures the fraction of documents correctly classified to the actual number of documents in the category.

$$\text{Recall} = x / (x + y'). \quad (4.2)$$

The error is the fraction of documents that are incorrectly assigned and incorrectly rejected from the particular category to the total number of documents in classification.

$$\text{Error} = (y + y')/(\text{Total Number of documents}) \quad , \quad (4.3)$$

where total number of documents= x+ y+ x' +y'.

The accuracy of the classifier is its ability to correctly assign the relevant documents and correctly reject the irrelevant documents from the particular category.

$$\text{Accuracy} = (x + x')/(\text{Total Number of documents}) \quad . \quad (4.4)$$

Precision becomes undefined when the classifier is not able to assign a category to particular documents [Yang 1991]. In such situations, recall becomes zero for particular cases. Thus, we calculated only the accuracy for the classifiers. In our research, we calculated the probability of classification error for each change category and made a

summation of all those classification errors to find the total classification error as shown below.

$$\text{Total Classification Error} = \sum_{i=1}^{n} P_i E_i \quad , \tag{4.5}$$

where n is the number of the categories in the classification, P_i is the probability of the documents in category i, and E_i is the classification error in category i.

In hierarchical classification, the classification error in *level-1* and *level-2* were combined to obtain the overall accuracy. The classification accuracy of the classifier was calculated as below.

$$\text{Total Classification Accuracy} = 1 - \text{Total Classification Error} \quad . \tag{4.6}$$

The classification accuracy obtained from equation (4.6) helped us to test the following hypothesis listed in Section 1.2.

Hypothesis #1: The automated change classification algorithms are effective to classify changes with high accuracy (> 80%).

4.3 Change Distribution

The NetBeans system and some systems from the Sun Developer Network (SDN) were selected in the study of change distributions: power-law and Weibull. Out of the 65 systems in the Sun Developer Network systems, only 4 systems met the requirements of the power-law test procedure and 19 for the Weibull test as discussed in Section 2.1.5.2. The system used test the power-law distribution were: NetBeans, Sun Java Studio, Java SE (JDK/JRE), Sun Studio, and Jini Network Technology. The system used for testing the Weibull distribution were: NetBeans, Sun Java Studio, Java SE (JDK/JRE), Sun Studio, Jini Network Technology, Sun Java System

Portalserver, Portal Server - Mobile Access, Java Enterprise Edition, Hotspot, Jiro, Java Web Services Developer Pack, Java Advanced Imaging, Portal Server – Search, Personal and Embedded Java, Java Plugin, Java Message Queue, Mobile Information Device Profile and Optional Package JSRS - Phoneme Feature, Java API for Xml-Based RPC, Java Web Server, and Mobile Information Device Profile Reference Implementation.

The bug reports in the bug repository of the NetBeans and SDN repositories were grouped to give a folder structure in each open source system and were referred to as products/components and category/sub-category respectively. We used the general term *module* hereafter to refer to the group of bug reports that were grouped into the same folder (file structure). Some of the modules in the bug repositories mapped to the source code modules. However, there might not always be the direct name to name matching between the modules in the bug repository and the source code modules. We used the summary information of those changesets in the changelog (discussed in Section 2.1.6) that contained bug ID, to map bug ID in the bug repository. The bug ID in the summary field of changeset in changelog was used to identify the source code module changed, while the bug ID in the bug repository was used to identify the module in bug repository. Furthermore, we also used the resolved bug ID descriptions and patches from the bug repository to identify the source code modules.

Besides the examination of the power-law and the Weibull distribution at the modules in the bug repository, we also performed the Pareto-Principle on the changed modules at the source code modules level. The data required for the test of the Pareto-Principle was obtained from the cluster study.

4.3.1. Procedure to Test the Power-Law Distribution

Mathematically, the power law distribution was represented by the equation: $p(x) \propto x^{-\alpha}$, where x is the input vector data and α *is* the constant scaling parameter. In our research, x is the number of change counts in each module (available in Appendix A). The number of modules is the number of observations for the power-law. The number of observations in the power-law must be greater than 50 in order to obtain a reliable result, as suggested by Clauset and colleagues [Clauset et al. 2009]. They provided a Matlab [Matlab2011] program to determine whether the given empirical data followed the power-law distribution or not.

The Matlab functions *plfit(), plplot()* and *plpva()* were used in this thesis to determine the *p-value* for the power-law distribution of the change data.
The function *plfit()* fits the power-law distributional model to the empirical data. It determines α as the maximum likelihood estimate of the scaling exponent, and *xmin* as the estimate of the lower bound of the power-law behavior. For the value of x greater or equal to *xmin*, the power- law is observed in the data. The function *plplot()* visualizes the power-law distributional model with empirical data based on x, *xmin* and α. The function *plpva()* takes the data x and *xmin* as the arguments and computes the corresponding goodness of fit test *p-value* for the fitted power-law model. A goodness-of-fit test value determines if the empirical data confirms to the hypothesized distribution. The generated *p-value* by *plpva()* function measures the distance between the distribution of the empirical data (input vector data) and the fitted power-law model (hypothesized) model. The fitted power-law model contained the synthetic (generative) data taking the *xmin* and the number of observation. Our empirical data followed the

hypothesized model data if 1 in 20 or less data agree. If the resulting *p-value* (power-law) is greater than 0.05 (threshold value), then it confirmed that the given data followed the power-law distribution. The function *plpva()* is an overloaded function and takes the number of repetitions as the argument. The default number of repetitions is 1000 in the tool, but for our purpose we set the number of repetitions to 2500 to bring the *p-value* (power-law) to be accurate to about two decimal digits after the decimal point.

The procedure to test the power-law distribution helped us to test the following hypotheses listed in Section 1.2.

Hypothesis #2: The total changes at the module level followed the power-law distribution.

Hypothesis #3: The error changes at the module level followed the power-law distribution.

Hypothesis #4: The non-error changes at the module level followed the power-law distribution.

The null hypotheses for the Hypotheses #2, #3 and #4 are stated as below:

Null Hypothesis for Hypothesis #2:

H0: The total changes at the module level did not follow the power-law distribution.

Null Hypothesis for Hypothesis #3:

H0: The error changes at the module level did not follow the power-law distribution.

Null Hypothesis for Hypothesis #4:

H0: The non-error changes at the module level did not follow the power-law distribution.

The stated null hypotheses were rejected if the *p-value* (power-law) was greater than 0.05. If the null hypotheses get rejected, then our particular change data (total changes, error changes, non-error changes) followed power-law distribution.

4.3.2. Procedure to Test the Weibull Distribution

For the test of the Weibull distribution, Minitab Software [Minitab 2011] was used. This software outputted the goodness of fit test *p-value*, taking the change data (available in Appendix A) as input. If the *p-value* is greater than 0.05 (threshold value), we confirmed the Weibull distribution in our data.

The procedure to test the Weibull distribution helped us to test the following hypotheses listed in Section 1.2.

Hypothesis #5: The total changes at the module level followed the Weibull distribution.

Hypothesis #6: The error changes at the module level followed the Weibull distribution.

Hypothesis #7: The non-error changes at the module level followed the Weibull distribution.

The null hypotheses for the Hypotheses #5, #6 and #7 are stated as below:

Null Hypothesis for Hypothesis #5:

H0: The total changes at the module level did not follow the Weibull distribution.

Null Hypothesis for Hypothesis #6:

H0: The error changes at the module level did not follow the Weibull distribution.

Null Hypothesis for Hypothesis #7:

H0: The non-error changes at the module level did not follow the Weibull distribution.

The stated null hypotheses were rejected if the *p-value* (Weibull) was greater than 0.05. It implies that our particular change data (total changes, error changes, non-error changes) followed the Weibull distribution.

4.4 Cluster Study

We used the NetBeans change data to perform the cluster analysis because it was the only system that provided the needed data for the study. The Cluster study was conducted in two phases: data collection of files changed and cluster analysis. The data collection of files changed is presented in Appendix C.

4.4.1. Cluster Analysis

An input transaction database is a collection of logical units of database operations on a set of items. The H-mine algorithm [Pei et al. 2007] output the complete set of frequent sub-patterns of items involved in the transaction. In our study, the changesets associated with a number of files represented the input transaction database for the H-mine algorithm and the algorithm output was the subset of files that changed frequently as shown in Appendix C (Figure C.3). The calculation of all the subsets of files affected by all 192,122 NetBeans changesets was not a trivial task. It required several days of computation and a very large hard disk space to store the results. Considering the computation and storage requirements, we set the *minsup* — the algorithm's support threshold [Pei et al. 2007], value to 21 for practical reasons. A *minsup* value of 21 tells the H-mine algorithm to find the subsets of files that were changed together 21 times or more.

For the purpose of the cluster analysis of software modules, we utilized the output of the H-mine algorithm, i.e., the subsets of files. We determined the software modules affected by subsets of files in each cardinality group. The information gathered from this procedure helped us to test the following hypothesis listed in Section 1.2.

Hypothesis #9: The most frequently changed file clusters tend to concentrate in a small set of modules.

CHAPTER 5

DATA ANALYSIS

In this chapter, we present the results of our research work.

5.1 Change Classification Results

This section depicts the results of manual and automated change classifications for the classes_lang package of Java SE (JDK/JRE) and Java Server Faces Technology.

5.1.1. Manual Classification Results

The results of manual classification for the two products are tabulated in Tables 5.1, 5.2, 5.3, and 5.4.

Table 5.1: Manual Classification Results for Package classes_lang

Change Category	Change Count	Overall Percentage
Concurrency	64	1.91
Configuration	59	1.76
Documentation	336	10.07
Interface	290	8.69
Internationalization	4	0.11
Maintenance	1365	40.92
Memory	29	0.86
Miscellaneous	36	1.07
Semantic	950	28.48
Standards	19	0.56
Test Bug	183	5.48
Total	3335	100

Table 5.2: Manual Classification (Sub Categories) Results for Package classes_lang

Change Category	Change Sub Category	Change Count	Percentage	Total
Semantic	Conceptual	290	30.52	950
	Exception	194	20.42	
	Functional	466	49.05	
Maintenance	Adaptive	91	6.67	1365
	Enhancement	1235	90.47	
	Refactoring	39	2.85	

Table 5.3: Manual Classification Results for Java Server Faces Technology

Change Category	Change Count	Overall Percentage
Concurrency	6	1.41
Configuration	11	2.58
Documentation	45	10.56
Interface	39	9.51
Internationalization	10	2.35
Maintenance	83	19.48
Miscellaneous	21	4.93
Semantic	196	46.01
Standards	15	3.52
Total	426	100

Table 5.4: Manual Classification (Sub Categories) Results for Java Server Faces Technology

Change Category	Change Sub Category	Change Count	Percentage	Total
Semantic	Conceptual	23	11.73	196
	Exception	39	19.89	
	Functional	134	68.36	
Maintenance	Adaptive	3	3.61	83
	Enhancement	55	66.26	
	Refactoring	25	30.12	

The column *Change Category* represents the category of the change description (Table 4.1). For change categories, Semantic and Maintenance are further classified into sub categories listed in the column *Change Sub Category.* The number of change descriptions that fell into the particular change category is listed in the *Change Count* column. The *Percentage* column represents the fraction of all documents that fell in a particular change category. The summation of all the change descriptions (change counts) in each change category is the total number of change data for a particular system.

5.1.2. Automated Classification Results

Automated change classification was performed in two rounds of iterations: supervised and random selection, for each product.

1. The classes_lang package

The classification results for package classes_lang in supervised selection are tabulated in Tables 5.5, 5.6, and 5.7. In these tables, column *Change Category* represents the category of the change description (Table 4.1). The *Training Set* column represents the number of documents taken from the particular change category to train the classifiers. The *Sample Set* column represents documents to be classified. The *Classifiers Results* column contains the results of the four classifiers Support Vector Machine (SVM), Naïve Bayes, Maximum Entropy (MaxEnt), and N-gram classifier. The number of bug Reports assigned by each classifier to a particular category are listed in column *Assigned* and the number of documents that were correctly classified is listed in column *Correct*.

Table 5.5: Flat Classification Results for the Package classes_lang

Change Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Concurrency	45	19	31	10	68	15	120	21	83	19
Configuration	41	18	16	10	29	13	27	13	46	18
Interface	247	43	742	208	546	193	463	167	533	188
Internationalization	2	2	0	0	0	0	1	0	0	0
Memory	19	10	6	6	13	7	13	8	20	8
Standards	12	7	1	1	2	1	1	1	2	1
Conceptual	264	26	100	64	127	77	188	94	144	60
Exception	184	10	10	10	37	33	43	34	59	38
Functional	433	33	341	190	425	221	391	198	360	194
Total	1247	168	1247	499	1247	560	1247	536	1247	526

Table 5.6: Hierarchical Classification Results for the Package classes_lang

Change Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Concurrency	45	19	4	1	40	7	78	16	48	13
Configuration	41	18	13	10	19	12	19	11	33	15
Interface	247	43	216	129	350	168	235	137	322	151
Internationalization	2	2	0	0	0	0	0	0	0	0
Memory	19	10	5	5	11	7	5	4	11	8
Semantic	881	69	1008	810	826	697	909	744	832	692
Standards	12	7	1	1	1	1	1	1	1	1
Total	1247	168	1247	956	1247	892	1247	913	1247	880

Table 5.7: Hierarchical (Semantic) Classification Results for Package Classes_lang

Change Sub Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Conceptual	264	26	187	95	216	111	295	135	244	101
Exception	184	10	9	9	80	53	63	42	82	57
Functional	433	33	685	373	585	343	523	306	555	326
Total	881	69	881	477	881	507	881	483	881	484

The same training set in Table 5.6 for the Semantic category (that was further classified into sub-categories) was taken to form the training set in Table 5.7. The flat classification results and hierarchical classification results for the package classes_lang in random selection are presented in Tables 5.8, 5.9, and 5.10.

Table 5.8: Flat Classification Results for Package classes_lang

Change Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Concurrency	55	9	0	0	1	1	6	2	9	6
Configuration	48	11	2	0	11	8	14	9	14	9
Interface	249	41	248	133	419	172	280	149	395	146
Internationalization	4	0	0	0	0	0	0	0	0	0
Memory	28	1	0	0	0	0	1	1	1	1
Standards	17	2	1	1	1	1	1	1	1	1
Conceptual	258	32	141	64	227	95	283	122	184	81
Exception	175	19	53	48	62	52	102	72	122	85
Functional	413	53	802	335	526	250	560	267	521	235
Total	1247	168	1247	581	1247	579	1247	623	1247	564

The training set count zero in the Internationalization change category indicates no training data was selected for this category. This change category was excluded by the classifier. A change category with low counts has a lower probability of being included in the training corpus and hence has a higher probability of being excluded from the change classification process.

Table 5.9: Hierarchical Classification Results Package classes_lang

Change Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Concurrency	55	9	0	0	1	1	0	0	6	4
Configuration	48	11	2	0	6	3	8	5	11	8
Interface	249	41	84	64	229	124	167	112	192	95
Internationalization	4	0	0	0	0	0	0	0	0	0
Memory	28	1	0	0	0	0	1	1	1	1
Semantic	846	104	1160	829	1010	769	1070	806	1036	766
Standards	17	2	1	1	1	1	1	1	1	1
Total	1247	168	1247	894	1247	898	1247	925	1247	875

Table 5.10: Hierarchical (Semantic) Classification Results for Package classes_lang

Change Sub Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Conceptual	258	32	138	80	259	129	246	129	205	108
Exception	175	19	44	40	84	69	82	66	133	100
Functional	413	53	664	374	503	318	518	327	508	313
Total	846	104	846	494	846	516	846	522	846	521

2. Java Server Faces Technology

The flat and hierarchical classification results for Java Server Faces Technology in the supervised selection are tabulated in Tables 5.11, 5.12, and 5.13.

Table 5.11: Flat Classification Results for Java Server Faces Technology

Change Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Concurrency	4	2	0	0	0	0	0	0	0	0
Configuration	7	4	0	0	0	0	0	0	0	0
Interface	30	9	0	0	7	4	1	1	4	3
Internationalization	7	3	0	0	1	0	1	0	2	0
Standards	10	5	0	0	0	0	1	0	1	0
Conceptual	16	7	0	0	12	0	5	0	2	1
Exception	25	14	5	5	19	13	13	9	23	17
Functional	99	35	193	99	159	83	177	90	166	91
Total	198	79	198	104	198	100	198	100	198	112

Table 5.12: Hierarchical Classification Results for Java Server Faces Technology

Change Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Concurrency	4	2	0	0	0	0	0	0	0	0
Configuration	7	4	0	0	0	0	0	0	0	0
Interface	30	9	0	0	4	2	1	1	3	2
Internationalization	7	3	0	0	0	0	0	0	1	0
Semantic	140	56	198	140	194	139	197	140	194	138
Standards	10	5	0	0	0	0	0	0	0	0
Total	198	79	198	140	198	141	198	141	198	140

Table 5.13: Hierarchical (Semantic) Classification Results for Java Server Faces Technology

Change Sub Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Conceptual	16	7	0	0	11	0	4	0	3	1
Exception	25	14	5	5	18	13	9	7	21	17
Functional	99	35	135	99	111	84	127	93	116	93
Total	140	72	140	104	140	97	140	100	140	111

The flat classification results and hierarchical classification results for Java Server Faces Technology in the random selection are tabulated in Tables 5.14, 5.15, and 5.16.

Table 5.14: Flat Classification Results for Java Server Faces Technology

Change Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Concurrency	6	0	0	0	0	0	0	0	0	0
Configuration	7	4	0	0	1	0	0	0	0	0
Interface	33	6	0	0	3	3	3	3	4	4
Internationalization	9	1	0	0	0	0	0	0	0	0
Standards	11	4	0	0	0	0	0	0	1	0
Conceptual	18	5	0	0	0	0	0	0	0	0
Exception	23	16	10	9	25	11	15	10	27	13
Functional	91	43	188	91	169	87	180	91	166	84
Total	198	79	198	100	198	101	198	104	198	101

The four classifiers could not recognize the Concurrency category because it was already excluded from the training i.e., the training set count was zero.

Table 5.15: Hierarchical Classification Result for Java Server Faces Technology

Change Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Concurrency	6	0	0	0	0	0	0	0	0	0
Configuration	7	4	0	0	1	0	0	0	0	0
Interface	33	6	0	0	2	2	2	2	2	2
Internationalization	9	1	0	0	0	0	0	0	0	0
Semantic	132	64	198	132	195	132	196	132	196	132
Standards	11	4	0	0	0	0	0	0	0	0
Total	198	79	198	132	198	134	198	134	198	134

Table 5.16: Hierarchical (Semantic) Classification Result for Java Server Faces Technology

Change Sub Category	Sample Set	Training Set	Classifiers Results							
			SVM		Naïve		MaxEnt		N-Gram	
			Assigned	Correct	Assigned	Correct	Assigned	Correct	Assigned	Correct
Conceptual	18	5	0	0	1	0	0	0	0	0
Exception	23	16	9	8	17	11	12	11	22	13
Functional	91	43	123	91	114	86	120	91	110	84
Total	132	64	132	99	132	97	132	102	132	97

The data presented in the *Sample Set* (documents to be classified) column, *Assigned* (documents assigned) column, and *Correct* (documents correctly assigned)

column in Tables 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5,14, 5.15, and 5.16 were used to calculate the applicable performance measures (accuracy) [Yang 1991] of the classifiers in our research. The methodology to calculate the accuracy for the classifiers was discussed in Section 4.2.3.5. The classification accuracy results of the classifiers in flat and hierarchical classification are tabulated in Tables 5.17 and 5.18 respectively. The overall accuracy of the hierarchical classification is presented in Table 5.19.

Table 5.17: Flat Classification Accuracy Measures of the Classifiers

Product	Iteration Plan	Classifiers Accuracy Measures			
		SVM	Naïve	MaxEnt	N-Gram
Package classes_lang	Supervised	0.74	0.76	0.75	0.75
	Random	0.75	0.76	0.78	0.76
Java Server Faces Technology	Supervised	0.71	0.72	0.71	0.75
	Random	0.72	0.74	0.74	0.74

Table 5.18: Hierarchical Classification Accuracy Measures of the Classifiers

Product	Iteration Plan	Classifiers Accuracy Measures							
		SVM		Naïve		MaxEnt		N-Gram	
		Level 1	Level 2	Level 1	Level 2	Level 1	Level 2	Level 1	Level 2
Package classes_lang	Supervised	0.81	0.66	0.78	0.6	0.79	0.67	0.77	0.67
	Random	0.77	0.69	0.79	0.7	0.80	0.72	0.77	0.71
Java Server Faces Technology	Supervised	0.76	0.78	0.77	0.7	0.77	0.76	0.77	0.82
	Random	0.74	0.69	0.75	0.7	0.75	0.72	0.75	0.71

Table 5.19: Overall Hierarchical Classification Accuracy for Classifiers

Product	Iteration Plan	Classifiers Accuracy Measures			
		SVM	Naïve	MaxEnt	N-Gram
Package classes_lang	Supervised	0.72	0.73	0.73	0.72
	Random	0.75	0.76	0.77	0.76
Java Server Faces Technology	Supervised	0.81	0.78	0.79	0.84
	Random	0.82	0.81	0.83	0.81

In Tables 5.17, 5.18 and 5.19, the classification accuracy of the automated classification was calculated by taking the results of the manual classification as the benchmark. The *Classification Accuracy Measures* column lists the accuracies obtained from the different classifiers in flat and hierarchical classification, taking the iteration plan into account. Iteration plan: supervised and random, was discussed in Section 4.2.3.3. The columns *Level 1* and *Level 2* in Table 5.18 represents the accuracy results obtained in *level-1* and *level-2* classification (discussed in Section 4.2.3.2.) respectively.

The classification accuracies of hierarchal classification are always higher than the flat classification at the same level (first attempt) of classification. The higher accuracy was the result of less change categories in the hierarchical (*level-1*) classification as compared to change categories in the flat classification. There was maximum 6% accuracy increase in the *level-1* classification. In *level-2* classification, Maximum Entropy (MaxEnt) classifier achieved 80% accuracy for the classification of the classes_lang package in the random selection. The N-Gram classifier achieved 82% accuracy for Java Server Faces Technology. We noticed that there was no significant

improvement in classification accuracy even though the hierarchical classification was applied.

5.3 Testing the Hypotheses

In this section, we present the results of testing the hypotheses listed in Section 1.2.

Hypothesis #1: The automated change classification algorithms are effective to classify changes with high accuracy (> 80%).

This hypothesis determines the usability of the automated classification with acceptable accuracy. Four classifiers, Support Vector Machine [Thorsten 2010], Naïve Bayes [McCallum 2002], MaxEnt [McCallum 2002], and N-Gram classifiers [Alias-i 2008], were used one at a time to classify the changes for the classes_lang package of Java SE (JDK/JRE) and the Java Sever Faces Technology. We presented the results of the automated classification in Section 5.1.2. The summarized results of flat and hierarchical classification are tabulated in Tables 5.17, 5.18 and 5.19.

On observation of the results in Table 5.17, we did not achieve accuracy greater than 80% in any of the software products using the flat classification. In Table 5.19, the overall accuracy of hierarchical classification for the package classes_lang was below 80%. Although, supervised hierarchical and random classification of the product Java Server Faces Technology was greater than 80%, the classification accuracies in all other cases were below 80%. This result implied Hypothesis #1 did not hold.

Based on our study and observation, the lack of a set of standard terms in the bug reports contributed to the inability of the classifiers to accurately group bug reports to the

right categories. We suggest that the open source community use standard terms to describe each bug report.

Hypothesis #2: The total changes at the module level followed the power-law distribution.

For the test of this hypothesis, we applied the procedure to test the power-law distribution described in Section 4.3.1. Out of the change data (Appendix A) in NetBeans and 65 Sun Developer Network systems, only five systems met the requirements for the power-law distribution test. The power-law distribution is valid when the number of modules in each system is greater than 50. Table 5.20 shows the five systems and their corresponding *p-value* (power-law) in the tests. As described in Section 4.3.1, if the *p-value* (power-law) was greater than 0.05, the null hypothesis gets rejected, and then the given data followed the power-law distribution. The *p-value* (power-law) for all the systems was greater than 0.05 and the null hypothesis for total changes were rejected. Hence, we affirmed Hypothesis #2 for the systems observed. The power-law distribution plot for the total change for each system is presented in Appendix B.

Table 5.20: Power-Law distribution Statistics for the Total Changes

System Name	Number of Modules	Power-Law p-value
NetBeans	530	0.3220
Sun Java Studio	141	0.4272
Java SE (JDK/JRE)	87	0.5176
Sun Studio	79	0.3860
Jini Network Technology	69	0.9492

Hypothesis #3: The error changes at the module level followed the power-law distribution.

To test this hypothesis, we applied a similar procedure in testing Hypothesis #2. Table 5.21 presents the test results for the error changes. The threshold *p-value* (power-law) for the power-law distribution was 0.05. The tests confirmed the power-law distribution (i.e., rejection of null hypothesis) for the error changes in all systems but for Jini Network Technology system we failed to reject the null hypothesis. Thus, Hypothesis #3 was observed for 4 in 5 (80%) systems. Our conclusion was the error changes at the module level generally followed power-law distribution. The power-law distribution plot for error changes for each system is presented in Appendix B.

Table 5.21: Power-Law distribution Statistics for the Error Changes

System Name	Number of Modules	Power-Law p-value
NetBeans	529	0.4288
Sun Java Studio	140	0.6008
Java SE (JDK/JRE)	86	0.4228
Sun Studio	73	0.2948
Jini Network Technology	65	0.0280

Hypothesis #4: The non-error changes at the module level followed the power-law distribution.

Table 5.22 summarizes the test results for the non-error changes. The module count of Sun Studio system was less than 50, thus no *p-value* (power-law) was calculated for it. The *p-value* (power-law) for the Jini Network Technology system was less than the threshold value of 0.05 so we failed to reject the null hypothesis. Hypothesis #4 hold

for only 3 (75%) in 4 systems. Our conclusion was: the non-error changes generally followed the power-law distribution in the observed systems. The power-law distribution plot for non-error changes for each system is presented in Appendix B.

Table 5.22: Power-Law distribution Statistics for the Non-Error Changes

System Name	Number of Modules	Power-Law p-value
NetBeans	491	0.4272
Sun Java Studio	117	0.8352
Java SE (JDK/JRE Package)	76	0.2548
Sun Studio	39	-
Jini Network Technology	53	0.0016

Hypothesis #5: The total changes at the module level followed the Weibull distribution.

The procedure to test the Weibull distribution was discussed in Section 4.3.2. We confirmed Hypothesis #5 based on the p-value (Weibull) greater than 0.05. In Table 5.23 *System Name* column list the name of the 20 software systems (NetBeans and 19 systems from Sun Developer Network) taken to observe the Weibull distribution. The *Number of Modules column* lists the modules count for each system. The *Weibull p-value* column lists the *p-value (Weibull)* calculated for each system. As we examined the *p-value* (Weibull) for the 20 systems listed, we obtained mixed results: some systems followed the Weibull distribution (rejection of null hypothesis for total changes) and some did not. The total changes for 16 (80%) systems followed the Weibull distribution and 4 systems did not. Our conclusion was: the total changes generally followed the Weibull distribution. The Weibull distribution plot for total changes for each system is presented in Appendix B.

Table 5.23: Weibull distribution Statistics for the Total Changes

System Name	Number of Modules	Weibull p-value
NetBeans	530	<0.05
Sun Java Studio	141	>0.05
Java SE (JDK/JRE)	87	>0.05
Sun Studio	79	<0.05
Jini Network Technology	69	<0.05
Sun Java System Portalserver	48	>0.05
Portal Server - Mobile Access	33	<0.05
Java EE (Enterprise Edition)	31	>0.05
Hotspot	26	>0.05
Jiro	25	>0.05
Java Web Services Developer Pack	25	>0.05
Java Advanced Imaging (JAI)	22	>0.05
Portal Server - Search	21	>0.50
Personal And Embedded Java	21	>0.50
Java Plugin	20	>0.05
Java Message Queue	20	>0.05
Mobile Information Device Profile and Optional Package JSRS - Phoneme Feature	20	>0.05
Java Api For Xml-Based Rpc	19	>0.05
Java Web Server	18	>0.05
Mobile Information Device Profile (MIDP) Reference Implementation	18	>0.05

Hypothesis #6: The error changes at the module level followed the Weibull distribution.

To test this hypothesis, the same systems from Table 5.23 were taken and the same procedure for the test of Hypothesis #5 was applied. The test results are summarized in Table 5.24. In Table 5.24, error changes for 16 (80%) systems followed the Weibull distribution (i.e., rejection of null hypothesis for error changes) and error changes for 4 systems did not follow the Weibull distribution. Our conclusion was the error changes at the module level generally followed the Weibull distribution. The Weibull distribution plot for error changes for each system is presented in Appendix B.

Table 5.24: Weibull distribution Statistics for the Error Changes

System Name	Number of Modules	Weibull p-value
NetBeans	530	<0.05
Sun Java Studio	141	>0.05
Java SE (JDK/JRE)	87	>0.05
Sun Studio	79	<0.05
Jini Network Technology	69	<0.05
Sun Java System Portalserver	48	>0.05
Portal Server - Mobile Access	33	<0.05
Java EE (Enterprise Edition)	31	>0.05
Hotspot	26	>0.05
Jiro	25	>0.05
Java Web Services Developer Pack	25	>0.05
Java Advanced Imaging (JAI)	22	>0.05
Portal Server - Search	21	>0.50
Personal And Embedded Java	21	>0.50
Java Plugin	20	>0.05
Java Message Queue	20	>0.05
Mobile Information Device Profile and Optional Package JSRS - Phoneme Feature	20	>0.05
Java Api For Xml-Based RPC	19	>0.05
Java Web Server	18	>0.05
Mobile Information Device Profile (MIDP) Reference Implementation	18	>0.05

Hypothesis #7: The non-error changes at the module level followed the Weibull distribution.

To test this hypothesis, the same systems from Table 5.23 were taken and the same procedure for the test of Hypothesis #5 was applied. Table 5.25 summarizes the test results for the non-error changes. Non-error changes for 8 systems (40%) followed the Weibull distribution (i.e., rejection of null hypothesis for total changes) and the remaining 12 systems did not. Our conclusion was: the non-error changes at the module level generally did not follow the Weibull distribution. The Weibull distribution plot for error changes for each system is presented in Appendix B.

Table 5.25: Weibull distribution Statistics for the Non-Error Changes

System Name	Number of Modules	Weibull p-value
NetBeans	530	<0.05
Sun Java Studio	141	<0.05
Java SE (JDK/JRE)	87	<0.05
Sun Studio	79	<0.05
Jini Network Technology	69	<0.05
Sun Java System Portalserver	48	>0.05
Portal Server - Mobile Access	33	<0.05
Java EE (Enterprise Edition)	31	<0.05
Hotspot	26	>0.05
Jiro	25	<0.05
Java Web Services Developer Pack	25	<0.05
Java Advanced Imaging (JAI)	22	>0.50
Portal Server - Search	21	>0.05
Personal And Embedded Java	21	>0.05
Java Plugin	20	>0.05
Java Message Queue	20	<0.05
Mobile Information Device Profile and Optional Package JSRS - Phoneme Feature	20	>0.05
Java Api For Xml-Based Rpc	19	<0.05
Java Web Server	18	>0.05
Mobile Information Device Profile (MIDP) Reference Implementation	18	<0.05

In Table 5.26, we summarized the results of testing the same systems, for which the power-law and the Weibull distribution were tested. We showed whether the same system followed the power-law and the Weibull distribution for total changes, error changes and non-error changes. The field with *NA* represents the *Not Applicable*, i.e., the data did not fulfill the requirements of the particular distribution test. The field with the value *Yes* indicates the particular type of change followed the particular distribution and the value *No* indicates the particular type of the change did not flow the particular distribution.

Table 5.26: Power-Law and Weibull Distribution Comparison on Same Systems

System Name	Power-Law			Weibull		
	Total Change	Error Change	Non-Error Change	Total Change	Error Change	Non-Error Change
NetBeans	Yes	Yes	Yes	No	No	No
Sun Java Studio	Yes	Yes	Yes	Yes	Yes	No
Java SE (JDK/JRE	Yes	Yes	Yes	Yes	Yes	No
Sun Studio	Yes	Yes	NA	No	No	No
Jini Network Technology	Yes	No	No	No	No	No

There were more systems that did not show the Weibull distribution than the power-law distribution. Our research results suggest that more studies are needed to conclude that the Weibull distribution is a better fit for software changes.

Hypothesis #8: The non-error changes were concentrated in the same modules where there was a high concentration of error changes.

To test the relationship between the non-error changes and error changes, we calculated the Spearman correlation coefficients and obtained Spearman critical values from [Conover 1980] to test the hypothesis. We used the Spearman correlation because we were not certain that the data were normally distributed. We test if the Spearman coefficient value was significant. The null hypothesis was stated as below:

H0: There is no relationship between the non-error changes and error changes.

Table 5.27 lists the Spearman correlation coefficients in the *Spearman* column, the number of modules in the *Module* column, the Spearman critical value in the *Spearman Critical Value* column and the test of the null hypothesis in the *Null Hypothesis* column.

The null hypotheses were rejected for 19 out of 20 systems tested. Our conclusion was the non-error changes generally concentrated in the same modules where there was a high concentration of error changes.

Table 5.27: Correlation Statistics and Null Hypothesis

System Name	Spearman Coefficient	Number of Module	Spearman Critical Value	Null Hypothesis
NetBeans	0.897	530	0.071	Rejected
Sun Java Studio	0.924	141	0.139	Rejected
Java SE (JDK/JRE)	0.919	87	0.177	Rejected
Sun Studio	0.461	79	0.186	Rejected
Jini Network Technology	0.771	69	0.199	Rejected
Sun Java System Portalserver	0.855	48	0.239	Rejected
Portal Server – Mobile Access	0.861	33	0.290	Rejected
Java EE	0.790	31	0.300	Rejected
Hotspot	0.840	26	0.329	Rejected
Jiro	0.417	25	0.336	Rejected
Java Web ServicesDeveloper Pack	0.642	25	0.336	Rejected
Java Advanced Imaging	0.666	22	0.359	Rejected
Portal Server-Search	0.872	21	0.368	Rejected
Personal and Embedded Java	0.759	21	0.368	Rejected
Java Plugin	0.769	20	0.378	Rejected
Java Message Queue	0.767	20	0.378	Rejected
Mobile Information Device Profile and Optional Package JSRS – Phoneme Feature	0.747	20	0.378	Rejected
Java Api For Xml- Based RPC	0.350	19	0.389	Rejection Failed
Java Web Server	0.931	18	.3994	Rejected
Mobile Information Device Profile (MIDP) Reference Implementation	0.713	18	.3994	Rejected

Hypothesis #9: The most frequently changed file clusters tend to concentrate in a small set of modules.

The H-mine algorithm outputted the subsets of files that were changed together. The changed files are contained in source code modules and thus the modules changed with the source files. Considering the computation and storage requirements, we set the *minsup* — the algorithm's support threshold [Pei et al. 2007], value to 21 for practical reasons because when the *minsup* value was set to 20, the H-mine algorithm took nearly 57 hours to run (for a Quad CPU with 2.66 GHz processor) and required 283 GB (a text file having subsets of files per line in the range of 10^{10} records) of hard disk space. In addition, inserting the subsets of files in the range 10^{10} records into database would take another 57 hours and 283 GB hard disk space or more. The *minsup* value determines the frequency of subsets of files that were changed together. We set the *minsup* value to 21 for optimization, which tells the H-mine algorithm to find the subsets of files that were changed together 21 times or more The lower the *minsup* value, the longer it takes the algorithm to run. Thus, we used the *minsup* value of 21, which took 886 milliseconds to run and required 77KB of hard drive space. We were interested in the subsets of files with cardinality greater or equal to 2 that were generated by the H-mine algorithm because a single file doesn't constitute a cluster. The number of modules affected by the frequently changed files for *minsup* value 21 is presented in Figure 5.1 below.

Figure 5.1 revealed the decreasing trend of the number of the module changed when the cardinality of the frequently changing subset of file increases. We calculate the percentage of modules where the subsets of files with cardinalities ranging from 2 to 9 were located.

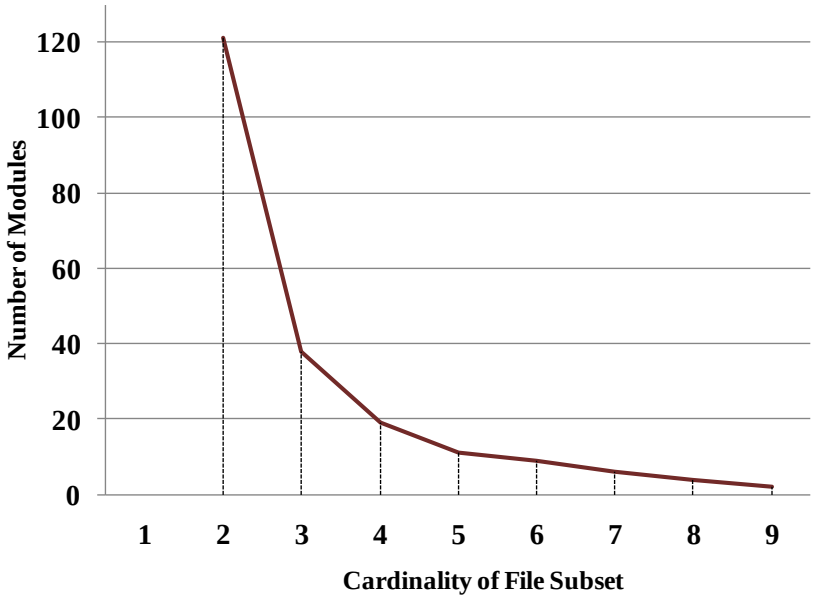

Figure 5.1: Number of Modules Changed versus File Subset Cardinality

Among 781 source code modules in NetBeans repository, files in 121 modules changed together to form different subsets with different cardinality.

The percentage of the most vulnerable modules (modules that changed due to the subset of files that changed with frequency greater or equal to 21) was calculated as below:

Percentage of modules changed =

$$\frac{\text{Number of modules changed}}{\text{Total number of modules}} = \frac{121}{781} = 15.59\,\%\,.$$

Within the sight of these results, frequently changed files concentrated in a small set of modules (less than 16%). Thus, we affirmed Hypothesis #9. The names of the affected modules are listed in Table 5.28. The *Subset Size* column represents the cardinality of the subset of files. The *Module Name* columns list the names of the modules that were changed by the subsets of files.

Table 5.28: Cardinality of Subset of Files with the Modules Name

Subset Size	Module Name
9	cnd.modelimpl , debugger.jpda
8	cnd.modelimpl , debugger.jpda, lib.profiler.charts, lib.profiler.ui
7	cnd.modelimpl, cnd.remote, debugger.jpda, cnd.completion, cnd.makeproject, lib.profiler.charts, lib.profiler.ui
6	cnd.completion, cnd.makeproject, cnd.modelimpl, cnd.remote, debugger.jpda, ide.ergonomics, lib.profiler, lib.profiler.charts, lib.profiler.ui
5	cnd.apt, cnd.completion, cnd.debugger.gdb, cnd.makeproject, cnd.modelimpl, cnd.remote, debugger.jpda, ide.ergonomics, lib.profiler, lib.profiler.charts, lib.profiler.ui
4	apisupport.project, asm, bpel.mapper, cnd.apt, cnd.completion, cnd.debugger.gdb, cnd.discovery, cnd.highlight, cnd.makeproject, cnd.model.services, cnd.modelimpl, cnd.remote, debugger.jpda, dlight.nativeexecution, dlight.remote, ide.ergonomics, lib.profiler, lib.profiler.charts, lib.profiler.ui
3	ant.debugger, apisupport.project, asm, bpel.mapper, bugtracking.bridge, bugzilla, cnd, cnd.api.model, cnd.apt, cnd.completion, cnd.debugger.gdb, cnd.discovery, cnd.editor, cnd.highlight, cnd.makeproject, cnd.model.services, cnd.modelimpl, cnd.remote, core.output2, debugger.jpda, dlight.annotationsupport, dlight.nativeexecution, dlight.remote, ide.ergonomics, java.editor, java.source, lib.profiler, lib.profiler.charts, lib.profiler.ui, masterfs, mercurial, performance, php.editor, subversion, versioning.system.cvss, visualweb.designer, visualweb.designer.jsf, web.jsf.navigation
2	ant.debugger, api.debugger, api.debugger.jpda, api.gsf, api.java.classpath, apisupport.project, asm, autoupdate.services, autoupdate.ui, bpel.core, bpel.editors, bpel.mapper, bugtracking, bugtracking.bridge, Bugzilla, cnd, cnd.api.model, cnd.api.remote, cnd.apt, cnd.completion, cnd.debugger.common2, cnd.debugger.gdb, cnd.debugger.gdb2, cnd.discovery, cnd.dwarfdiscovery, cnd.dwarfdump, cnd.editor, cnd.gizmo, cnd.highlight, cnd.lexer, cnd.makeproject, cnd.model.services, cnd.modeldiscovery, cnd.modelimpl, cnd.remote, cnd.utils, compapp.casaeditor, core.output2, core.startup, core.windows, css.editor, db, db.dataview, db.metadata.model, db.sql.editor, debugger.jpda, debugger.jpda.ui, dlight.annotationsupport, dlight.collector.stdout, dlight.core.stack, dlight.db.derby, dlight.db.h2, dlight.derby.support, dlight.dtrace, dlight.management,
2	dlight.nativeexecution, dlight.perfan, dlight.remote, dlight.remote.impl, dlight.visualizers, editor, editor.lib, editor.lib2, form, form.j2ee, glassfish.common, glassfish.javaee, groovy.editor, groovy.grails, html.editor, html.editor.lib, ide.ergonomics, j2ee.clientproject, j2ee.earproject, j2ee.ejbjarproject, 2ee.persistence, j2ee.weblogic9, j2eeserver, java.editor, java.hints, java.source, jira, kenai, kenai.ui, lib.profiler, lib.profiler.charts, lib.profiler.ui, masterfs, maven.jaxws, mercurial, nbbuild, o.n.bootstrap, o.n.core, openide.explorer, openide.nodes,parsing.api, performance, php.editor, php.project, profiler, properties, ruby, ruby.testrunner, subversion, uihandler, versioning.system.cvss, visualweb.designer, visualweb.designer.jsf, vmd.components.svg, vmd.midp, web.client.tools.common, web.core.syntax, web.jsf, web.jsf.editor, web.jsf.navigation, web.project, websvc.kit, websvc.rest, websvc.restapi, xml.multiview, xml.schema.completion

Hypothesis # 10: Approximately 80% of changes were concentrated in about 20% of the changed modules (Pareto Principle).

This hypothesis is related to the popular 80/20 rule, also known as the Pareto Principle [Pareto 1906]. The total number of atomic changes (changes in a single java source file) recorded for the NetBeans main repository was 259,100. This accounted for the total number of changes made to all files in the NetBeans repository. The files were contained in source modules. As a result of a change in files, the source modules that contained the files were also affected. There were 621 modules which were affected by the total changes, while the Java files in the remaining 160 modules were untouched.

Figure 5.2 illustrates the distribution of changes in the source code modules which were affected by the total changes. Figure 5.2 is a plot between the module frequency arranged in descending order with the corresponding changed modules. The frequencies of modules changed are presented in Appendix D.

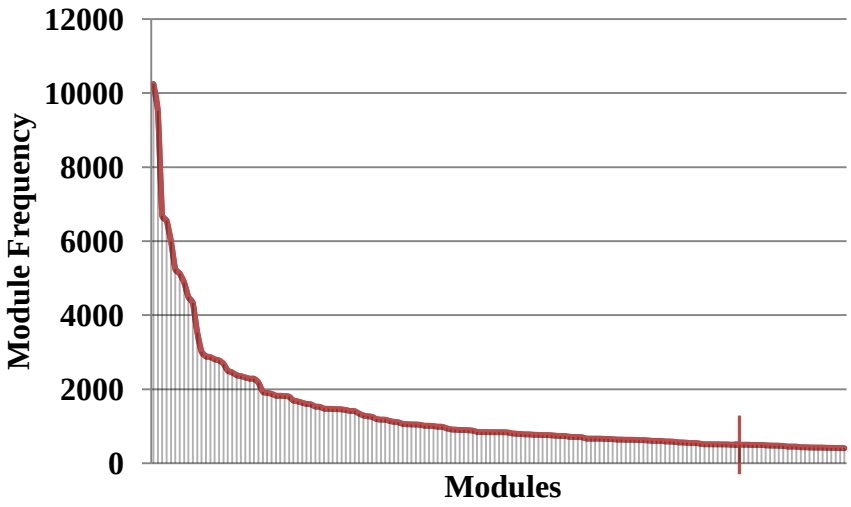

Figure 5.2: Distribution of Changes in Modules Changed

The cumulative percentage of frequencies of the changed modules with the changed number of modules is shown in Figure 5.3.

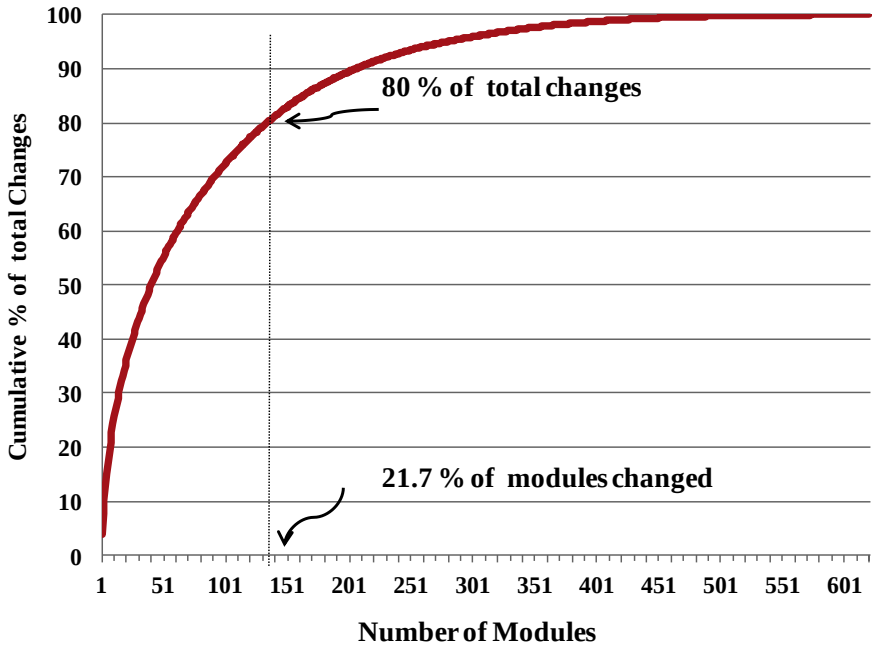

Figure 5.3: Plot of the Pareto Principle for the Changed Modules

For this hypothesis to be true, we must have the following results:

80% of the changes= total changes * 0.8= 259100* 0.8=207,280.

20% of the modules= total modules changes * 0.2= 621*0.2=124.2= 125 [number of modules cannot be a fraction].

This implies that 80% of the total changes must occur in 125 changed modules. In Figure 5.3, 80% of the changed module meets the horizontal axis between the values 130 and 140. But in our research, 80% of changes were recorded in 135 modules, i.e., 21.7% of the total changed modules. Thus, we affirmed this hypothesis and we concluded that the Pareto Principle applied.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we have investigated software changes from several open source system in three disciplines: classification study, distribution study, and clustering study. The experiments carried out on manual and automated change classifications led to the following conclusions.

1. Automated classification of software changes was not accurate. The inaccuracy was largely contributed by the limited text describing the nature of the changes and the lack of standard terms used in describing changes. Our research results suggest that using standard terms in documenting open source system changes will improve the accuracy of automated change classification.
2. Creation of training data through random selection was not reliable because it excluded some of the change categories with low change counts.
3. The automated classification accuracy was increased when the number of change categories was reduced.
4. Some change categories showed dominant behavior over the others. For example, in our data, the Maintenance category accounted for around 20% to 40% of all changes requests logged in the bug repositories. The Semantic category followed

the Maintenance category in dominance behavior and accounted for 28% to 46% of all changes logged.

Our study on change distribution involved the test of the power-law and the Weibull distribution in the change data. The following conclusions were drawn:

1. The total changes at the module level followed the power-law distribution.
2. The error changes in most of the systems followed the power-law distribution.
3. The non-error changes in most of the systems followed the power-law distribution.
4. The total changes for most of the systems followed the Weibull distribution.
5. The error changes for most of the systems followed the Weibull distribution.
6. The non-error changes for most of the systems did not follow the Weibull distribution.
7. For most systems, the non-error changes were concentrated in the same modules where there was a high concentration of error changes.

In the clustering study, the changed software source modules were identified from the changed source files. The most frequently changed files concentrated in a small set of modules. The subsets of files with a change frequency greater than 20 occurred in about 16% of the modules.

6.1 Future Work

In the future, we seek to improve and explore more in our research work. This includes the architectural redesign of our SDN web crawler tool. The crawler will be implemented in a multi-threaded fashion that will reduce the harvesting time in web by utilizing the CPU time better.

The four classifiers (Support Vector Machine, Naïve Bayes, Maximum Entropy and N-Gram) used in our research for automated change classification did not provide the acceptable classification accuracy. Future work includes the exploration of other types of classification tools and techniques to improve the classification accuracy of the automated classification.

During the clustering study, the lines changed information from the NetBeans main repository was collected for only the java source files. Future work includes the collection of lines changed information for other non-java files which will allow us to identify the frequently changing modules due to non-java files also. We performed optimization in terms of the computation time and the hard disk memory while finding the cluster of files that change together. In our results, we observed the decreasing trends in the module set cardinality when the file subset cardinality increases. Future work in this area includes the exploration of this trend when the *minsup* value is lowered than 21. Based on our research findings, we believe that a prediction model can be developed to predict the small set of modules in arbitrary systems that are prone to changes.

APPENDICES

APPENDIX A

ERROR CHANGE AND NON ERROR CHANGE DATA

A. NetBeans Project

The non-Error changes include Request for Enhancement RFE) and Task in NetBeans system.

Table A.1: Error and Non-Error Changes for NetBeans Version Control Product

Version Control				
Component	**Error**	**RFE**	**Task**	**Total**
CVS	1728	334	23	2085
CVS library	113	24	7	144
Code	327	85	10	422
Subversion	1851	244	10	2105
Localhistory	117	16	1	134
Mercurial	709	97	7	813
ClearCase	140	17	2	159
Git	62	8	0	70
Total	5047	825	60	5932

Table A.2: Error and Non-Error Changes for NetBeans Gui Builder Product

Gui Builder				
Component	**Error**	**RFE**	**Task**	**Total**
Natural Layout	484	36	0	520
Code	4010	528	62	4600
App Framework	418	30	1	449
Binding	157	24	1	182
Total	5069	618	64	5751

Table A.3: Error and Non-Error Changes for NetBeans Update Centers Product

Update Centers				
Component	**Error**	**RFE**	**Task**	**Total**
AU Masters	94	11	8	113
Beta	53	0	126	179
Pluginportal	193	69	0	262
Stable	95	6	101	202
Dev	77	2	25	104
Third-party	10	0	28	38
Hotfixes	3	0	7	10
Total	525	88	295	908

Table A.4: Error and Non-Error Changes for NetBeans Mobility Product

Mobility				
Component	**Error**	**RFE**	**Task**	**Total**
Visual Designer	1106	154	3	1263
Build System	483	82	16	581
Web Services	189	16	2	207
CDC pack	274	24	1	299
-- Other --	569	103	12	684
Game Builder	95	21	1	117
JUnit	74	7	1	82
Fragmentation	38	31	1	70
Total	2828	438	37	3303

Table A.5: Error and Non-Error Changes for NetBeans Debugger Product

Debugger				
Component	**Error**	**RFE**	**Task**	**Total**
Java	2281	206	33	2520
Code	2666	490	90	3246
Ant	99	5	0	104
Total	5046	701	123	5870

Table A.6: Error and Non-Error Changes for NetBeans Editor Product

Editor				
Component	**Error**	**RFE**	**Task**	**Total**
-- Other --	4281	650	141	5072
Macros	80	39	1	120
Hints & Annotations	401	106	4	511
Search	192	89	3	284
Lexer	309	13	28	350
Refactoring	1476	226	17	1719
Completion &	1047	255	7	1309
Painting & Printing	750	61	6	817
Code folding	296	76	7	379
Actions/Menu/Toolbar	233	73	4	310
Settings	297	36	10	343
Key bindings	271	59	5	335
Options	671	73	3	747
Navigation	184	84	2	270
Formatting &	380	134	4	518
CSL (API &	328	23	2	353
Parsing & Indexing	514	32	12	558
Spellchecker	80	15	0	95
Total	11790	2044	256	14090

Table A.7: Error and Non-Error Changes for NetBeans Java Product

Java				
Component	**Error**	**RFE**	**Task**	**Total**
Javadoc	746	152	34	932
Unsupported	4747	451	312	5510
I18N	336	73	12	421
Classfile	48	11	3	62
Project	1675	437	25	2137
JUnit	763	151	24	938
Editor	1656	485	10	2151
Classpath	133	23	14	170
Beans	356	66	7	429
Platform	245	50	1	296
Source	2462	177	44	2683
Hints	1072	201	4	1277
Navigation	540	111	3	654
Freeform	200	55	5	260
Compiler	648	16	5	669
Refactoring	1220	149	5	1374
Total	16847	2608	508	19963

Table A.8: Error and Non-Error Changes for NetBeans Profiler Product

Profiler				
Component	**Error**	**RFE**	**Task**	**Total**
Base	2295	469	70	2834
Engine	47	0	0	47
Ide	56	8	0	64
Attach	1	1	1	3
DTrace	8	4	0	12
Total	2407	482	71	2960

Table A.9: Error and Non-Error Changes for NetBeans Obsolete Product

Obsolete				
Component	**Error**	**RFE**	**Task**	**Total**
collabnet	1529	318	337	2184
apisupport	86	21	8	115
jarpackager	435	39	39	513
cpplite	53	5	3	61
vcscore	946	124	98	1168
vcscvs	229	39	0	268
jndi	58	3	3	64
jasm	7	1	2	10
rmi	129	14	24	167
applet	66	15	3	84
xtest	215	98	15	328
vcsgeneric	1491	192	45	1728
corba	123	15	23	161
antlr	7	0	23	30
jini	12	3	1	16
sim	114	19	11	144
externaleditor	83	10	0	93
treefs	31	4	1	36
languages	523	66	5	594
archivesupport	5	3	2	10
serialversion	9	4	0	13
EJB Freeform	58	4	1	63
collab	251	49	4	304
metrics	22	7	3	32
visualweb	3255	484	59	3798
classclosure	5	0	1	6
form-swingx	7	1	0	8
icebrowser	95	6	0	101
makefile	6	0	0	6
debuggertools	48	1	0	49
netbrowser	2	0	3	5
cpp	61	11	4	76
wasp	2	0	3	5
webl	10	1	0	11
form-jndc	17	1	0	18
crosscompile	5	0	1	6
innertesters	5	1	0	6
javaembeddedserver	2	0	0	2
Total	10002	1559	722	12283

Table A.10: Error and Non-Error Changes for NetBeans Soa Product

Soa				
Component	**Error**	**RFE**	**Task**	**Total**
BPEL	980	157	4	1141
BPEL Project	351	43	0	394
JBI Test Driver	141	15	9	165
JBI Manager	142	22	2	166
BPEL Debugger	234	44	11	289
BPEL Mapper	441	95	3	539
-- Other --	162	21	8	191
Composite	556	166	4	726
BPEL Validation	158	43	1	202
Refactoring	28	3	0	31
XSLT	369	60	3	432
Binding	156	11	3	170
IEP editor	184	33	2	219
IEP project	58	1	2	61
SQL Project	110	2	0	112
OpenESB addons	15	3	0	18
File BC	33	1	0	34
JMS BC	33	1	0	34
WorkList Manager	50	8	1	59
Database BC	20	2	0	22
HTTP BC	19	2	0	21
Cobol Copy Book	8	0	0	8
REST BC	7	1	0	8
Data Integrator	20	3	0	23
FTP BC	12	2	0	14
LDAP BC	12	1	0	13
Scheduler BC	7	7	0	14
Mashbups	1	0	0	1
HL7 BC	24	3	0	27
Total	4331	750	53	5134

Table A.11: Error and Non-Error Changes for NetBeans Db Product

Db				
Component	**Error**	**RFE**	**Task**	**Total**
Code	1223	326	45	1594
SQL Editor	324	108	3	435
Derby	191	38	5	234
DB schema	128	19	3	150
Show Data	268	53	1	322
MySQL	185	23	3	211
Total	2319	567	60	2946

Table A.12: Error and Non-Error Changes for NetBeans JavaScript Product

Javascript				
Component	**Error**	**RFE**	**Task**	**Total**
Debugger	269	21	0	290
Editor	557	81	1	639
Libraries	36	4	2	42
-- Other --	25	3	2	30
JSON	6	2	0	8
Refactoring	14	0	0	14
EJS	2	0	0	2
jMaki	5	0	0	5
Monitor	3	0	0	3
Components	2	1	0	3
Total	919	112	5	1036

Table A.13: Error and Non-Error Changes for NetBeans Cnd Product

Cnd				
Component	**Error**	**RFE**	**Task**	**Total**
Terminalemulator	232	32	6	270
Code	1172	122	14	1308
Code folding	23	5	0	28
Project	874	181	7	1062
Code Model	1425	95	25	1545
Debugger	933	128	5	1066
Classview	96	7	0	103
Code Completion	376	31	2	409
Qnavigator	26	3	0	29
Editor	479	79	7	565
Accessibility	85	5	1	91
-- Other --	339	59	4	402
I18N	13	0	0	13
Navigation	118	33	1	152
Remote	711	52	3	766
D-Light	311	13	8	332
sh	16	3	0	19
Profile	273	21	8	302
execution	161	11	3	175
Toolchain	58	26	1	85
ASM	43	30	6	79
Total	7764	936	101	8801

Table A.14: Error and Non-Error Changes for NetBeans IDE Product

IDE				
Component	**Error**	**RFE**	**Task**	**Total**
Code	2253	289	60	2602
Internal Server	128	14	11	153
Report Exception	428	53	9	490
Accessibility	263	36	2	301
Registration	93	6	0	99
Import Settings	166	30	11	207
Logger	237	22	1	260
BlueJ	71	8	0	79
Features On	251	7	2	260
Commit Validation	38	0	0	38
Welcome	218	25	5	248
UI	380	128	81	589
Extbrowser	261	38	21	320
Schema2Beans	96	22	5	123
libs	28	13	9	50
Logger Server	39	7	0	46
Slowness Detector	20	2	0	22
Manifest	5	0	0	5
Edu	3	0	0	3
Total	4978	700	217	5895

Table A.15: Error and Non-Error Changes for NetBeans Installer Product

Installer				
Component	**Error**	**RFE**	**Task**	**Total**
Code	1368	177	71	1616
NBI	484	79	137	700
Mac Native	90	14	1	105
Tools bundle	114	2	8	124
JavaFX	19	1	1	21
SUN Studio	52	23	9	84
JDK bundle	132	3	22	157
Solaris Packages	10	0	0	10
Debian	24	3	0	27
RPM	11	2	1	14
Total	2304	304	250	2858

Table A.16: Error and Non-Error Changes for NetBeans Web service Product

Webservice				
Component	**Error**	**RFE**	**Task**	**Total**
Code	1189	143	18	1350
Client	221	17	3	241
JAX-WS	439	71	4	514
WSIT	365	43	1	409
REST	521	60	7	588
JAXB	51	27	1	79
Designer	125	10	1	136
JAX-RPC	138	4	0	142
Editor	29	13	0	42
Manager	303	14	2	319
Customization	31	6	0	37
SoapUI	23	0	0	23
ZWAG	5	0	0	5
Registr	19	4	1	24
Axis2	23	7	1	31
Total	3482	419	39	3940

Table A.17: Error and Non-Error Changes for NetBeans Utilities Product

Utilities				
Component	**Error**	**RFE**	**Task**	**Total**
Properties	387	42	10	439
Search	513	133	27	673
Image	52	19	9	80
Diff	419	97	22	538
Open File	88	36	4	128
Code	196	23	7	226
Group	7	4	4	15
URL	15	1	5	21
Print	84	19	1	104
PDF	23	2	0	25
Jump To	95	23	1	119
Test Runner	42	7	0	49
Total	1921	406	90	2417

Table A.18: Error and Non-Error Changes for NetBeans Xml Product

Xml				
Component	**Error**	**RFE**	**Task**	**Total**
Tools	97	37	5	139
Code	783	132	65	980
TAX	26	3	2	31
TAX/Lib	22	2	7	31
API	22	5	2	29
CSS	22	3	0	25
Text-Edit	296	63	12	371
Schema	108	25	2	135
Tree-Edit	63	31	2	96
Catalog	49	23	9	81
XSL	68	30	3	101
XML Multiview	54	8	2	64
Schema Model	85	11	1	97
Schema Tools	1112	139	35	1286
XDM	75	4	7	86
WSDL Tools	666	135	2	803
XAM	112	22	2	136
WSDL Model	94	2	0	96
Catalog Support	27	12	0	39
JAXB	54	2	0	56
Refactoring	32	1	1	34
Validation	38	11	1	50
Lexer	6	0	0	6
Document Model	24	1	0	25
XSLT Model	6	2	0	8
Retriever	24	5	2	31
Total	3965	709	162	4836

Table A.19: Error and Non-Error Changes for NetBeans web Product

web				
Component	**Error**	**RFE**	**Task**	**Total**
HTML	905	123	16	1044
CSS	517	50	6	573
YAML	47	3	0	50
Refactoring	39	3	0	42
Editing	20	0	0	20
Total	1528	179	22	1729

Table A.20: Error and Non-Error Changes for NetBeans Users guide Product

Usersguide				
Component	**Error**	**RFE**	**Task**	**Total**
Code	564	67	56	687
XML	30	3	4	37
Projects	23	4	19	46
Installer	7	3	3	13
Java	34	6	5	45
JavaEE	90	9	5	104
Ant	18	2	1	21
Database	19	0	3	22
API Support	60	8	2	70
SOA	152	6	0	158
CND	43	1	2	46
IDE	13	6	1	20
JavaFX	147	3	3	153
Editor	23	8	1	32
Form	16	3	1	20
Profiler	28	4	8	40
Web Services	20	1	1	22
Mobility	84	5	2	91
UML	37	5	11	53
Server Plugins	9	1	0	10
Ruby	8	7	0	15
PHP	13	1	1	15
Web tutorials	6	0	1	7
Utilities	1	1	0	2
Web	3	0	0	3
JavaScript	1	0	0	1
Python	7	0	0	7
Total	1456	154	130	1740

Table A.21: Error and Non-Error Changes for NetBeans Javaee Product

Javaee				
Component	**Error**	**RFE**	**Task**	**Total**
Code	3602	453	217	4272
HTTP Monitor	228	55	25	308
JSP	1524	130	27	1681
Debugger	328	22	2	352
Web Project	1284	126	9	1419
DD Editor	234	18	2	254
Refactoring	153	20	3	176
EAR	471	42	5	518
EJB	1284	101	8	1393
EJB Project	297	23	4	324
Web Freeform	119	12	1	132
JSP Parser	161	9	5	175
Struts	145	20	1	166
Palette	63	4	0	67
JSF	756	91	8	855
Persistence	981	137	17	1135
App Client	151	12	2	165
Editor	148	16	0	164
Facelets-1.1.x	39	6	0	45
Archiveproject	65	9	1	75
Spring	144	24	4	172
Hibernate	224	24	0	248
Maven	200	22	5	227
JSF Editor	249	21	3	273
CDI	17	6	0	23
Servlet	22	1	0	23
BluePrints	88	7	1	96
Wicket	11	0	0	11
Samples	27	3	0	30
Expression	13	0	0	13
GWT	11	0	0	11
App Engine	5	0	1	6
Total	13044	1414	351	14809

Table A.22: Error and Non-Error Changes for NetBeans Projects Product

Projects				
Component	**Error**	**RFE**	**Task**	**Total**
Generic	1678	276	226	2180
Generic Projects	1260	348	62	1670
Ant	1028	200	33	1261
Libraries	194	89	6	289
Ant Project	300	52	4	356
Eclipse project	157	29	6	192
Ant Freeform	127	38	4	169
JBuilder project	31	5	0	36
Java Webstart	92	24	5	121
Maven	1458	174	21	1653
Web Opener	52	7	1	60
JavaFX	237	36	14	287
Autoproject	30	24	0	54
Maven OSGi	17	4	2	23
Total	6661	1306	384	8351

Table A.23: Error and Non-Error Changes for NetBeans UML Product

UML				
Component	**Error**	**RFE**	**Task**	**Total**
General Diagram	697	105	0	802
Code Generation	245	44	8	297
General	857	103	22	982
Diagram Class	437	79	0	516
Reporting	92	25	0	117
Project	308	46	1	355
Reverse	295	49	1	345
Synchronization	167	9	1	177
Design Center	151	15	0	166
Diagram Sequence	283	25	1	309
Output	44	16	0	60
Diagram Activity	125	9	1	135
Properties	65	7	0	72
Diagram	13	5	0	18
Diagram	33	2	0	35
Diagram State	80	4	0	84
Diagram	22	1	0	23
Diagram Usecase	45	8	0	53
Total	3959	552	35	4546

Table A.24: Error and Non-Error Changes for NetBeans Contrib Product

Contrib				
Component	**Error**	**RFE**	**Task**	**Total**
Looks	52	16	10	78
Java3D	24	5	11	40
Filecopy	11	1	0	12
Naming	4	1	6	11
Tasklist	194	50	7	251
RemoteFS	48	14	4	66
Code	151	21	7	179
JMX	64	27	1	92
Quick Filechooser	16	1	0	17
PMD	18	4	0	22
Jackpot	236	13	2	251
Java Usages	7	0	0	7
File Search	6	2	1	9
DocBook	7	0	0	7
Portalpack	254	43	1	298
Latex	32	4	4	40
Scala	153	11	0	164
Codecoverage	22	7	2	31
Erlang	14	0	0	14
Selenium	11	1	0	12
Omnidebugger	1	2	1	4
TestNG	29	6	2	37
Sysprops	5	0	0	5
Workspace	23	2	0	25
Registry	6	7	11	24
Importcruncher	3	0	0	3
Module Manager	8	10	1	19
Perspective	2	0	0	2
Terminal	7	4	0	11
Smarty	30	5	0	35
Projectpackager	6	1	0	7
Jalopy	1	1	0	2
Buildmonitor	1	1	0	2
PlatformX	1	3	0	4
Flying Saucer CSS	1	0	0	1
Total	1448	263	71	1782

Table A.25: Error and Non-Error Changes for NetBeans Third-Party Product

Third-Party				
Component	**Error**	**RFE**	**Task**	**Total**
-- Other --	411	17	3	431
nb4openvms	7	0	1	8
xtt	1	0	0	1
nbxdoclet	10	0	0	10
sqe tools	53	1	0	54
DBX-Gui	156	3	6	165
Total	638	21	10	669

Table A.26: Error and Non-Error Changes for NetBeans Qa Product

Qa				
Component	**Error**	**RFE**	**Task**	**Total**
Code	505	88	72	665
a11y	56	1	1	58
Jellytools	152	22	21	195
Tests	258	17	22	297
Hudson4QE	18	4	0	22
Test Tools	11	1	0	12
Test Specifications	1	0	0	1
Total	1001	133	116	1250

Table A.27: Error and Non-Error Changes for NetBeans Apisupport Product

Apisupport				
Component	**Error**	**RFE**	**Task**	**Total**
Harness	325	74	7	406
Project	1341	223	28	1592
Templates	265	52	2	319
Inspector	12	11	0	23
API docs	9	3	2	14
Refactoring	43	6	0	49
Maven	64	7	0	71
Total	2276	388	54	2718

Table A.28: Error and Non-Error Changes for NetBeans Nblocalization Product

Nblocalization				
Component	**Error**	**Non-Error**	**Task**	**Total**
Code	217	12	15	244

Table A.29: Error and Non-Error Changes for NetBeans www Product

www				
Component	**Error**	**RFE**	**Task**	**Total**
Builds & Repositories	476	133	117	726
Admin	163	13	3	179
Obsolete	62	8	4	74
Downloads	37	1	0	38
Bugzilla	97	9	2	108
Web Content	59	5	4	68
dreamteam	0	1	1	2
Total	894	170	131	1195

Table A.30: Error and Non-Error Changes for NetBeans Performance Product

Performance				
Component	**Error**	**RFE**	**Task**	**Total**
Code	279	41	19	339
Tests	52	3	4	59
Timers	33	0	1	34
Total	364	44	24	432

Table A.31: Error and Non-Error Changes for NetBeans Groovy Product

Groovy				
Component	**Error**	**RFE**	**Task**	**Total**
Grails	242	63	6	311
Editor	247	35	21	303
GSP	21	2	1	24
Code	196	29	5	230
Total	706	129	33	868

Table A.32: Error and Non-Error Changes for NetBeans Php Product

Php				
Component	**Error**	**RFE**	**Task**	**Total**
Code	430	69	12	511
Debugger	230	31	1	262
Editor	1471	268	27	1766
Project	661	70	14	745
Refactoring	41	14	0	55
Symfony	35	10	0	45
Formatting &	289	16	1	306
FTP Support	104	47	1	152
Zend	27	2	1	30
PHPDoc	18	7	0	25
PHPUnit	84	21	1	106
Navigation	41	15	0	56
Ini files	3	0	0	3
Total	3434	570	58	4062

Table A.33: Error and Non-Error Changes for NetBeans Javafx Product

Javafx				
Component	**Error**	**RFE**	**Task**	**Total**
-- Other --	222	20	17	259
Editor	1068	100	61	1229
Deployment	86	9	3	98
Debugger	147	13	10	170
Navigator	24	1	2	27
Platform	54	4	4	62
Preview	88	8	4	100
Profiler	28	1	5	34
Palette	88	31	2	121
Composer	88	8	0	96
Refactoring	164	2	2	168
Parsing	34	0	0	34
FXD & FXZ	63	19	0	82
Total	2154	216	110	2480

Table A.34: Error and Non-Error Changes for NetBeans Python Product

Python				
Component	**Error**	**RFE**	**Task**	**Total**
-- Other --	30	6	1	37
Options	6	3	0	9
Debugger	66	3	0	69
Platform	15	2	1	18
Projects	68	11	1	80
Editor	125	20	0	145
Code	99	6	0	105
Testing	4	3	1	8
Build	4	1	0	5
Console	2	2	0	4
Navigation	5	0	0	5
Total	424	57	4	485

Table A.35: Error and Non-Error Changes for NetBeans Connected developer Product

Connecteddeveloper				
Component	**Error**	**RFE**	**Task**	**Total**
Hudson	107	45	2	154
Issuetracking	246	42	7	295
Kenai Framework	472	45	3	520
Bugzilla	281	38	3	322
Chat	159	12	0	171
Jira	121	16	3	140
Total	1386	198	18	1602

Table A.36: Error and Non-Error Changes for NetBeans Javacard Product

Javacard				
Component	**Error**	**RFE**	**Task**	**Total**
Java Card	140	21	5	166
Total	140	21	5	166

Table A.37: Error and Non-Error Changes for NetBeans Platform Product

Platform				
Component	**Error**	**RFE**	**Task**	**Total**
-- Other --	6149	538	250	6937
Window System	5767	821	448	7036
Autoupdate	1092	146	57	1295
Explorer&Property	1516	188	64	1768
Module System	660	118	27	805
Lookup	249	28	15	292
Data Systems	709	75	26	810
Nodes	369	50	14	433
Dialogs&Wizards	523	114	38	675
Text	599	79	15	693
Filesystems	1430	164	36	1630
Documentation	83	21	26	130
Help System	239	46	17	302
Actions	308	61	64	433
Options&Settings	529	68	9	606
Execution	194	27	13	234
Tasklist	354	55	31	440
Property Editors	200	54	7	261
Outline&TreeTable	337	37	4	378
NB JUnit	52	18	5	75
Favorites	124	24	4	152
Plugin Manager	759	173	13	945
Output Window	575	84	2	661
Progress	158	14	0	172
Launchers&CLI	113	11	1	125
Palette	218	35	4	257
Graph	197	80	8	285
Directory Chooser	161	13	1	175
Navigator	119	19	1	139
Quick Search	111	23	11	145
JDK Problems	130	0	0	130
Templates	32	9	0	41
Embedded Browser	88	12	1	101
Netigso	35	1	2	38
Plugin Importer	23	2	0	25
Total	24202	3208	1214	28624

Table A.38: Error and Non-Error Changes for NetBeans Ruby Product

Ruby				
Component	**Error**	**RFE**	**Task**	**Total**
Code	672	91	3	766
Debugger	251	34	18	303
Platform	139	22	9	170
Rails	311	71	7	389
Project	169	29	2	200
Refactoring	52	4	0	56
RHTML	98	13	1	112
Editing	308	61	4	373
Gems	120	42	0	162
Navigation	32	19	0	51
Code Completion	50	14	0	64
Testing	164	34	1	199
Rake	52	19	2	73
Hints	17	6	0	23
Total	2435	459	47	2941

Table A.39: Error and Non-Error Changes for NetBeans Serverplugins Product

Serverplugins				
Component	**Error**	**RFE**	**Task**	**Total**
Infrastructure	820	152	39	1011
Tomcat	804	109	31	944
Sun Appserver 8	604	31	3	638
Code	124	35	47	206
JBoss	329	59	4	392
WebLogic	195	38	1	234
WebSphere	118	13	0	131
Sun Appserver 9	808	69	22	899
AVK	13	0	1	14
Identity	195	11	2	208
WebServer 7	32	5	0	37
OC4J	38	6	4	48
GlassFish v3	566	73	35	674
Sun Webserver	31	5	1	37
Business One	1	0	0	1
Jetty	3	1	0	4
Total	4681	607	190	5478

B. Sun Products

Table A.40: Error and Non-Error Changes for Connected Limited Device Configuration (CLDC) – Specification

Connected Limited Device Configuration(CLDC) - Specification			
Package/Subcategory	**Error**	**Non-Error**	**Total**
Other-	5	1	6
jvm-compatibility	10	0	10
libraries-java	51	6	57
general	2	1	3
libraries-javax	96	3	99
Total	164	11	175

Table A.41: Error and Non-Error Changes for Connected Limited Device Configuration (CLDC)

Connected Limited Device Configuration(CLDC)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
documentation	1	1	2
debugger	87	0	87
libraries	25	0	25
vm	693	19	712
other	52	3	55
Total	858	23	881

Table A.42: Error and Non-Error Changes for Custom Doclets For The Javadoc Tool

Custom Doclets For The Javadoc Tool			
Package/Subcategory	**Error**	**Non-Error**	**Total**
mifdoclet	323	60	383
commentmergedoclet	2	1	3
localdoclet	7	1	8
stubmakerdoclet	6	0	6
doccheckdoclet	5	5	10
xmldoclet	1	4	5
Total	344	71	415

Table A.43: Error and Non-Error Changes for IDL Compiler

IDL Compiler			
Package/Subcategory	**Error**	**Non-Error**	**Total**
other	10	5	15
spec	1	1	2
java	44	4	48
c++	23	2	25
Total	78	12	90

Table A.44: Error and Non-Error Changes for Hotspot

Hotspot			
Package/Subcategory	**Error**	**Non-Error**	**Total**
jvm_interface	61	1	62
performance	1	1	2
compiler1	890	48	938
monitoring_management	67	2	69
jni	175	25	200
jvmti	622	68	690
test	1417	212	1629
jvmdi	401	6	407
hybrid_interpreter	64	10	74
machine_description	24	2	26
compiler2	4033	540	4573
mtask	51	10	61
runtime_system	2898	387	3285
attach	5	1	6
jvmpi	407	14	421
svc_agent	26	0	26
other	1211	122	1333
jkernel	2	3	5
solaris	161	3	164
tools	85	16	101
stub_generator	3	0	3
doc	23	4	27
client	130	1	131
garbage_collector	1803	467	2270
tools_jconsole	27	2	29
perfdata	14	1	15
Total	14601	1946	16547

Table A.45: Error and Non-Error Changes for Gcc For Sparc Systems

Gcc For Sparc Systems			
Package/Subcategory	**Error**	**Non-Error**	**Total**
compiler	13	0	13
Total	13	0	13

Table A.46: Error and Non-Error Changes for Inactive Categories

Inactive Categories			
Package/Subcategory	**Error**	**Non-Error**	**Total**
admin	9	1	10
ui	14	0	14
Total	23	1	24

Table A.47: Error and Non-Error Changes for Java Access Bridge For Assistive Technology

Java Access Bridge For Assistive Technology			
Package/Subcategory	**Error**	**Non-Error**	**Total**
other	8	0	8
accessbridge	127	21	148
at	8	0	8
doc	3	0	3
Total	146	21	167

Table A.48: Error and Non-Error Changes for Java Architecture For Xml Binding (JAXB-XSD)

Java Architecture For Xml Binding (JAXB-XSD)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
doc	40	3	43
spec	106	37	143
compiler	382	33	415
xjc_invalid_schemas	20	1	21
schemagen	28	1	29
runtime	190	24	214
Total	766	99	865

Table A.49: Error and Non-Error Changes for Java Access Bridge For Java Advanced Imaging (JAI)

Java Advanced Imaging (JAI)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
product_fcs	1	0	1
specification	22	36	58
codeclib	38	19	57
web_site	1	1	2
product_beta	25	0	25
test_framework	22	1	23
medialib	19	7	26
product_alpha	31	2	33
product_test	1	1	2
demo	4	2	6
installation	12	5	17
codec_imageio	186	69	255
codec_sample	57	23	80
reference_port	459	18	477
doc	142	10	152
applet	0	3	3
build_process	2	1	3
jai_1.0	49	4	53
other	16	9	25
implementation	218	28	246
jai_1.0.1	39	3	42
jai_1.1	266	14	280
Total	1610	256	1866

Table A.50: Error and Non-Error Changes for Java Authentication and Authorization Service (JAAS)

Java Authentication And Authorization Service (JAAS)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
other	18	9	27
callback	3	1	4
doc	5	1	6
module	25	3	28
login	22	8	30
auth	41	5	46
Total	114	27	141

Table A.51: Error and Non-Error Changes Java Api For Xml-Based RPC

Java API For Xml-Based Rpc (JAX-RPC)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
server-runtime	43	1	44
client-runtime	72	2	74
dii	49	0	49
wsdl-generator	36	1	37
serialization	39	0	39
interoperability	30	0	30
spec	19	3	22
encoding	12	2	14
configuration	2	1	3
serializer-generator	17	1	18
wsdl-parser	29	0	29
soap-package	2	0	2
streaming-parser	5	0	5
stub-generator	25	0	25
skeleton-generator	15	0	15
tie-generator	20	0	20
other	109	6	115
tools	119	11	130
schema-parser	34	1	35
Total	677	29	706

Table A.52: Error and Non-Error Changes for Java Business Integration (JBI)

Java Business Integration (JBI)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
framework	53	9	62
deploy_service	18	0	18
nms	10	0	10
common-client	2	0	2
cli	59	3	62
demo	26	1	27
wsdl	1	0	1
synchronization	1	0	1
management	131	7	138
install_service	18	1	19
admin_service	18	0	18
installer	57	1	58
ant_tasks	49	1	50
Total	443	23	466

Table A.53: Error and Non-Error Changes for Java Cryptography Extension

Java Cryptography Extension (JCE)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
other	15	2	17
sunmscapi	10	4	14
install	1	0	1
build	11	1	12
pkcs11_csp	153	24	177
runtime	112	20	132
classes_crypto	145	29	174
doc	64	6	70
aes_csp	10	1	11
Total	521	87	608

Table A.54: Error and Non-Error Changes for Java Data Objects

Java Data Objects (JDO)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
common	28	1	29
tck	13	1	14
query	12	3	15
enhancer	24	2	26
spec	3	0	3
javax	7	0	7
fostore	18	0	18
Total	105	7	112

Table A.55: Error and Non-Error Changes for Java Database Connectivity

Java Database Connectivity			
Package/Subcategory	**Error**	**Non-Error**	**Total**
jdbc_odbc_bridge	349	39	388
docs	3	0	3
eod	55	0	55
doc	7	0	7
other	2	0	2
spec	89	38	127
rowset_implementation	195	8	203
rowset_spec	139	19	158
implementation	333	68	401
Total	1172	172	1344

Table A.56: Error and Non-Error Changes for Java Generic Security Services

Java Generic Security Services (JGSS)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
nativeprovider	19	3	22
framework	19	7	26
krb5plugin	153	46	199
build	6	1	7
spnego	11	2	13
doc	15	0	15
Total	223	59	282

Table A.57: Error and Non-Error Changes for Java Deployment

Java Deployment			
Package/Subcategory	**Error**	**Non-Error**	**Total**
unittest	5	4	9
componentization	2	1	3
jkernel	314	41	355
deployment_toolkit	31	16	47
update	205	19	224
general	533	111	644
console	36	6	42
desktop	24	5	29
compression	51	8	59
download	77	21	98
security	93	37	130
networking	23	7	30
configuration	171	30	201
Total	1565	306	1871

Table A.58: Error and Non-Error Changes for Java Guides

Java Guides			
Package/Subcategory	**Error**	**Non-Error**	**Total**
none	3052	217	3269
Total	3052	217	3269

Table A.59: Error and Non-Error Changes for Java Management Extensions

Java Management Extensions (JMX)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
classes	507	162	669
other	21	4	25
tools	2	2	4
tck	26	4	30
doc	141	28	169
Total	697	200	897

Table A.60: Error and Non-Error Changes for Java IDL

Java IDL			
Package/Subcategory	**Error**	**Non-Error**	**Total**
tools	41	5	46
test	34	1	35
dynamic-typing1	29	1	30
orb	627	41	668
nameserver	52	6	58
orbd	23	1	24
poa	126	5	131
doc	28	1	29
transaction	8	0	8
transport	42	3	45
java-idl	13	2	15
security	6	0	6
serialization	67	5	72
other	151	18	169
rmic-iiop	68	7	75
idl-java	81	13	94
spec	20	3	23
Total	1416	112	1528

Table A.61: Error and Non-Error Changes for Java Management Extensions (JMX) Remote API

Java Management Extensions (JMX) Remote API			
Package/Subcategory	**Error**	**Non-Error**	**Total**
optional_classes	36	7	43
examples	8	0	8
other	13	0	13
Total	57	7	64

Table A.62: Error and Non-Error Changes for Java Enterprise Edition

Java EE (Enterprise Edition)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
jacc	1	0	1
webservices	22	0	22
j2ee-cx-spec	1	0	1
l10n_ja	2	0	2
cmp	38	0	38
repository	3	0	3
xml	12	0	12
test-bugs	2	0	2
verification	6	0	6
management	9	0	9
jmsra	27	0	27
deployment	557	31	588
entity_container	148	4	152
specification	33	9	42
protocols	32	2	34
deploytool	1046	85	1131
session_container	76	1	77
transactions	52	1	53
other	372	35	407
console	58	3	61
admin	68	6	74
security	149	6	155
web	127	2	129
naming	33	0	33
codegen	39	0	39
install	40	9	49
i18n	82	1	83
jms	37	4	41
connectors	41	4	45
doc	101	9	110
ejbql	35	0	35
Total	3249	212	3461

Table A.63: Error and Non-Error Changes for Java ME SDK

Java ME SDK			
Package/Subcategory	**Error**	**Non-Error**	**Total**
nbplatform	99	7	106
source_bundle	9	0	9
emulator	370	33	403
demos	163	9	172
toolbar	479	85	564
legal	1	0	1
uei	10	1	11
build	69	3	72
other	162	23	185
cdc	19	3	22
monitors	15	4	19
jsrop	27	0	27
cldc	12	0	12
installer	61	5	66
midp	14	0	14
doc	50	1	51
Total	1560	174	1734

Table A.64: Error and Non-Error Changes for Java ME Fragmentation

Java ME Fragmentation			
Package/Subcategory	**Error**	**Non-Error**	**Total**
midp	0	1	1
mmapi	0	1	1
other	2	2	4
Total	2	4	6

Table A.65: Error and Non-Error Changes for Java ME TCK Framework

Java ME TCK Framework			
Package/Subcategory	**Error**	**Non-Error**	**Total**
interview	9	1	10
opensrc	1	0	1
doc	7	0	7
other	4	2	6
src	15	13	28
Total	36	16	52

Table A.66: Error and Non-Error Changes for Java Media Framework

Java Media Framework			
Package/Subcategory	**Error**	**Non-Error**	**Total**
player	1487	51	1538
demo	37	2	39
test	20	1	21
audio	91	9	100
video	114	6	120
processor	19	4	23
rtsp	16	5	21
doc	11	2	13
api	23	13	36
rtp	58	1	59
install	29	2	31
capture	21	0	21
other	89	10	99
Total	2015	106	2121

Table A.67: Error and Non-Error Changes for Java Message Queue

Java Message Queue			
Package/Subcategory	**Error**	**Non-Error**	**Total**
c_api	59	41	100
demo	87	7	94
l10n_japan	4	0	4
ia_java_api	7	0	7
l10n	94	0	94
build	22	1	23
i18n	55	1	56
jmx	49	7	56
source	2	0	2
l10n-asia	20	0	20
windows	67	0	67
admin	345	70	415
jms_api	499	27	526
installation	296	6	302
test	405	38	443
other	197	37	234
jms_ra	93	14	107
broker	2028	238	2266
hp-ux	12	0	12
doc	194	10	204
Total	4535	497	5032

Table A.68: Error and Non-Error Changes for Java Naming and Directory Interface

Java Naming And Directory Interface (JNDI)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
classlib	15	7	22
ldap	233	22	255
cosnaming	15	2	17
doc	39	2	41
other	37	4	41
class	47	5	52
dns	33	6	39
rmi	6	1	7
nis	2	0	2
dsml	3	0	3
test_jndi	28	6	34
Total	458	55	513

Table A.69: Error and Non-Error Changes for Java Plugin

Java Plugin			
Package/Subcategory	**Error**	**Non-Error**	**Total**
doc	76	15	91
solaris	160	8	168
compatibility	105	0	105
applet_spec	26	21	47
linux	56	3	59
l10n	103	0	103
bridge	174	19	193
iexplorer	449	9	458
install	138	19	157
i18n	74	3	77
i10n	3	0	3
plugin2	747	77	824
plugin	1037	90	1127
ocx	400	19	419
converter	187	16	203
other	568	66	634
ns6	137	5	142
misc	741	70	811
plugin3	14	2	16
l18n	2	0	2
Total	5197	442	5639

Table A.70: Error and Non-Error Changes for Java SE Timezone Update Tool

Java SE Timezone Update Tool			
Package/Subcategory	**Error**	**Non-Error**	**Total**
tool	67	54	121
Total	67	54	121

Table A.71: Error and Non-Error Changes for Java SE Visualvm

Java SE Visualvm			
Package/Subcategory	**Error**	**Non-Error**	**Total**
tool	10	2	12
Total	10	2	12

Table A.72: Error and Non-Error Changes for Java Secure Socket Extension

Java Secure Socket Extension (JSSE)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
build	28	1	29
runtime	554	89	643
regtests	22	0	22
examples	16	0	16
doc	87	3	90
install	1	3	4
Total	708	96	804

Table A.73: Error and Non-Error Changes for Java SE (JDK/JRE)

Java SE (JDK/JRE)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
servlet	21	7	28
classes_security	1784	468	2252
javah	91	22	113
cte_docs	82	6	88
monitoring	52	26	78
sunservicetags	34	17	51
jwsse-nightly-bld	48	0	48
javap	119	52	171
performance	13	1	14
web_staging	62	0	62
exact	0	1	1

Table A.73 (cont'd…)

classes_other	1	0	1
test	53	7	60
classes_annot_processing	77	10	87
classes_util_jarzip	317	84	401
classes_text	1291	198	1489
rmi	867	180	1047
classes_beans	547	158	705
debugger	1405	179	1584
jit	211	18	229
classes_util_i18n	1560	556	2116
server	12	4	16
native_interface	263	35	298
classes_lang_model	51	12	63
jlf_design_guidelines	2	0	2
xml-parser	1	0	1
demo_2d	55	3	58
classes_2d	6664	626	7290
classes_dyn	18	4	22
classes_nio2	8	0	8
svc_demos	3	2	5
other	327	81	408
classes_awt	11277	942	12219
classes_java	259	13	272
build	1680	360	2040
compiler	3934	584	4518
runtime	2634	186	2820
mtask	41	2	43
jigsaw	26	5	31
cobundle-install	11	1	12
qa_web_eng	296	13	309
jvm_exact	28	0	28
jwsse-nightly-test	11	0	11
snmp	5	1	6
classes_util	1675	658	2333
classes_instrument	6	3	9
doc	1	0	1
appletviewer	375	37	412
staging	7	0	7
module	103	59	162
classes_awt_i18n	75	11	86
jit_symantec	402	2	404
imageio	578	99	677

Table A.73 (cont'd…)

apt	128	43	171
profiling	128	30	158
serviceability	479	119	598
classes_util_concurrent	282	56	338
accessibility	284	47	331
classes_lang	2350	985	3335
classes_math	99	50	149
classes_fontprop	390	66	456
jconsole	258	96	354
man	4	0	4
classes_io	1601	323	1924
classes_net	2671	372	3043
classes_management	173	26	199
specification	554	430	984
demo	198	19	217
cte_test	504	17	521
classes_nio	1032	136	1168
serialization	445	78	523
classes_swing	14255	1619	15874
jar	323	76	399
classes_awt_im	551	66	617
dragndrop	661	53	714
char_encodings	645	115	760
install	2751	384	3135
tools	371	117	488
classes_util_regex	160	59	219
javadoctool	232	94	326
idl	24	4	28
l10n-japan	36	3	39
extmech	69	16	85
localization	867	61	928
classes_util_logging	136	61	197
qa_install	39	1	40
classes_sound	805	154	959
Total	72968	11509	84477

Table A.74: Error and Non-Error Changes for Java Servlet

Java Servlet			
Package/Subcategory	**Error**	**Non-Error**	**Total**
http	99	9	108
spec	13	1	14
examples	3	0	3
admin	4	0	4
doc	124	11	135
api	99	45	144
other	72	14	86
internal_servlet	9	0	9
Total	423	80	503

Table A.75: Error and Non-Error Changes for Java Shared Data Toolkit

Java Shared Data Toolkit			
Package/Subcategory	**Error**	**Non-Error**	**Total**
other	1	1	2
http	2	1	3
Total	3	2	5

Table A.76: Error and Non-Error Changes for Java Tools Documents

Java Tools Documents			
Package/Subcategory	**Error**	**Non-Error**	**Total**
linux	1	0	1
windows	3	0	3
solaris	7	0	7
all	43	4	47
Total	54	4	58

Table A.77: Error and Non-Error Changes for Java Tutorial

Java Tutorial			
Package/Subcategory	**Error**	**Non-Error**	**Total**
none	25	7	32
Total	25	7	32

Table A.78: Error and Non-Error Changes for Javabeans Activation Framework

Javabeans Activation Framework			
Package/Subcategory	**Error**	**Non-Error**	**Total**
examples	1	1	2
classes	49	17	66
tests	5	0	5
doc	27	2	29
other	6	3	9
	88	23	111
Total	1	1	2

Table A.79: Error and Non-Error Changes for Java Web Server

Java Web Server			
Package/Subcategory	**Error**	**Non-Error**	**Total**
doc	125	4	129
proxy	33	2	35
disk_cache	1	0	1
performance	18	1	19
servlet_embedding	10	1	11
installation	56	5	61
session_tracking	30	2	32
java_server_pages	45	3	48
servlets	322	27	349
server_side_include	29	0	29
authstore	30	0	30
security	171	28	199
admin_back_end	176	9	185
log	48	9	57
http	231	12	243
other	475	35	510
page_compilation	73	5	78
admin_front_end	245	29	274
Total	2118	172	2290

Table A.80: Error and Non-Error Changes Java Web Start

Java Web Start			
Package/Subcategory	**Error**	**Non-Error**	**Total**
netbeans	22	2	24
jnlp_file	143	20	163
app_mgr	300	36	336
doc	67	8	75
maintenance	12	16	28
spec	35	59	94
download_engine	125	11	136
other	740	118	858
enterprise_support	3	2	5
demo	35	3	38
general	426	52	478
l10n	74	1	75
autodownload	42	3	45
i18n	39	8	47
install	168	19	187
jnlp_api	112	23	135
Total	2343	381	2724

Table A.81: Error and Non-Error Changes for Java3D

Java3D			
Package/Subcategory	**Error**	**Non-Error**	**Total**
input_si	2	0	2
vrml	29	1	30
sound_si	100	1	101
other	67	21	88
examples	37	1	38
doc	133	4	137
utilities	194	19	213
graphics_si	2286	22	2308
test	60	0	60
classes_vecmath	7	2	9
install	23	3	26
Total	2938	74	3012

Table A.82: Error and Non-Error Changes for Java Web Services Developer Pack

Java Web Services Developer Pack			
Package/Subcategory	**Error**	**Non-Error**	**Total**
docs	84	2	86
jaxb	12	0	12
ant	14	0	14
jaxp	10	0	10
registry_server	16	3	19
jstl	5	0	5
other	332	9	341
tools	127	4	131
tomcat	54	0	54
integration	123	1	124
samples	31	1	32
installer	181	13	194
jaxrpc	26	2	28
jaxr	22	2	24
saaj	12	0	12
xws-security	10	0	10
packages	5	1	6
wsi-sa	10	0	10
admintool	46	2	48
plugin	8	0	8
jes_registry	10	0	10
jax-fast	3	1	4
xmldsig	2	0	2
jax-ws	3	0	3
tutorial	6	1	7
Total	1152	42	1194

Table A.83: Error and Non-Error Changes for Javahelp

Javahelp			
Package/Subcategory	**Error**	**Non-Error**	**Total**
solarispkgs	32	0	32
search_engine	78	10	88
development	339	69	408
demos	57	5	62
spec	18	7	25
doc	23	2	25
other	119	20	139
Total	666	113	779

Table A.84: Error and Non-Error Changes for Javamail

Javamail			
Package/Subcategory	**Error**	**Non-Error**	**Total**
file	1	0	1
pop3	0	1	1
smtp	23	10	33
tests	6	0	6
other	54	15	69
imap	70	14	84
examples	7	1	8
generic	66	38	104
doc	13	1	14
internet	61	48	109
Total	301	128	429

Table A.85: Error and Non-Error Changes for Javaserver Faces Technology

Javaserver Faces Technology			
Package/Subcategory	**Error**	**Non-Error**	**Total**
portlet	1	0	1
jsfaces_demo	5	1	6
jsfaces_spec	32	4	36
jsfaces_docs	6	0	6
jsfaces_api	50	9	59
jsfaces_ri	298	20	318
Total	392	34	426

Table A.86: Error and Non-Error Changes for Javaserver Javaserver Pages Standard Tag Library (JSTL)

Javaserver Pages Standard Tag Library (JSTL)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
ri	7	1	8
spec	7	2	9
other	6	0	6
Total	20	3	23

Table A.87: Error and Non-Error Changes for Jax-WS

Jax-WS			
Package/Subcategory	**Error**	**Non-Error**	**Total**
handlers	43	0	43
wsimport	135	7	142
wsgen	58	2	60
interoperability	27	1	28
client-runtime	243	5	248
spi	2	0	2
j2se-endpoint	12	0	12
serializer-generator	4	1	5
server-runtime	171	11	182
test	1	0	1
other	186	27	213
tools	96	8	104
spec	20	6	26
Total	998	68	1066

Table A.88: Error and Non-Error Changes for JAXP

JAXP			
Package/Subcategory	**Error**	**Non-Error**	**Total**
xslt	296	13	309
dom3	60	5	65
dom	234	14	248
other	418	47	465
xml11	3	0	3
tutorial	2	0	2
stax	129	13	142
performance	5	7	12
parse	55	3	58
validation	66	3	69
xpath	39	8	47
spec	59	27	86
doc	35	1	36
sax	243	15	258
Total	1644	156	1800

Table A.89: Error and Non-Error Changes for Jini Network Technology

Jini Network Technology			
Package/Subcategory	**Error**	**Non-Error**	**Total**
com_sun_jini_action	0	2	2
net_jini_export	3	1	4
net_jini_constraint	1	0	1
net_jini_id	1	1	2
com_sun_jini_reggie	76	48	124
net_jini_io	1	0	1
com_sun_jini_logging	2	4	6
net_jini_activation	9	1	10
net_jini_url	2	4	6
csj_qa_spec_discoveryservice	2	0	2
com_sun_jini_benchmark_outrigger	2	0	2
csj_qa_impl_start	4	1	5
com_sun_jini_proxy	2	0	2
com_sun_jini_admin	1	0	1
net_jini_discovery	62	20	82
com_sun_jini_tool	52	34	86
com_sun_jini_debug	3	1	4
csj_test_lookup	11	0	11
csj_tck_test	9	3	12
csj_qa_impl_mercury	0	1	1
csj_test_misc	2	1	3
csj_qa_impl_outrigger	2	0	2
csj_qa_spec_lookupservice	3	0	3
com_sun_jini_outrigger	246	49	295
other	35	10	45
net_jini_lookup	103	19	122
doc	61	22	83
build	26	8	34
net_jini_lease	18	9	27
net_jini_core	48	28	76
net_jini_event	2	2	4
com_sun_jini_norm	39	6	45
com_sun_jini_config	2	2	4
com_sun_jini_fiddler	57	4	61
com_sun_jini_phoenix	25	7	32
com_sun_jini_mercury	57	13	70
com_sun_jini_thread	21	16	37
net_jini_iiop	3	0	3
net_jini_jrmp	5	0	5
net_jini_security	10	6	16
net_jini_entry	8	0	8

Table A.89 (cont'd…)

csj_test_util	2	0	2
csj_test_txnmanager	13	2	15
csj_qa_harness	5	20	25
com_sun_jini_benchmark	1	1	2
csj_test_discovery	1	0	1
com_sun_jini_reliablelog	5	1	6
com_sun_jini_mahout	6	2	8
csj_test_space	46	9	55
csj_test_mailbox	3	2	5
com_sun_jini_collection	0	1	1
com_sun_jini_constants	4	0	4
com_sun_jini_system	1	0	1
com_sun_jini_lease	12	6	18
com_sun_jini_lookup	7	1	8
com_sun_jini_start	43	29	72
com_sun_jini_mahalo	96	17	113
net_jini_space	17	2	19
com_sun_jini_example	65	29	94
net_jini_loader	25	10	35
net_jini_jeri	134	52	186
csj_tck_harness	20	6	26
csj_harness	2	2	4
csj_tck_naughtydog	2	1	3
csj_tck_admin	2	2	4
csj_qa_share	0	1	1
net_jini_config	16	16	32
com_sun_jini_discovery	11	9	20
com_sun_jini_benchmark_lookup	5	1	6
Total	1560	545	2105

Table A.90: Error and Non-Error Changes for JT Harness (JavaTest)

JT Harness (JavaTest)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
utils	4	0	4
build	1	1	2
tests	0	1	1
doc	42	9	51
core	158	64	222
Total	205	75	280

Table A.91: Error and Non-Error Changes for Jiro

Jiro			
Package/Subcategory	**Error**	**Non-Error**	**Total**
station	19	0	19
jarpackw	4	0	4
igniter	39	2	41
viewdbg	1	0	1
viewlog	10	5	15
jarpack	8	2	10
securadm	1	0	1
jarx	1	0	1
jiroadm	2	0	2
securmgr	2	0	2
security	2	0	2
tutorials	2	0	2
jiroc	4	3	7
documents	14	0	14
jardeploy	0	1	1
igniterw	14	1	15
transaction_manager	3	1	4
other	56	11	67
compliance	2	0	2
log_service	11	2	13
event_service	10	0	10
installation	13	3	16
proxygen	0	1	1
scheduling_service	4	0	4
jirocw	2	1	3
Total	224	33	257

Table A.92: Error and Non-Error Changes for K Virtual Machine

K Virtual Machine			
Package/Subcategory	**Error**	**Non-Error**	**Total**
libraries	113	12	125
vm	389	28	417
specifications	44	5	49
documentation	5	0	5
other	45	17	62
debugger	92	6	98
Total	688	68	756

Table A.93: Error and Non-Error Changes for JAXR

JAXR			
Package/Subcategory	**Error**	**Non-Error**	**Total**
browser	21	2	23
other	47	4	51
infomodel	46	1	47
client	7	1	8
doc	14	2	16
spec	42	5	47
interop	6	0	6
communication	2	0	2
uddireg	1	0	1
uddi_provider	141	2	143
Total	327	17	344

Table A.94: Error and Non-Error Changes for Mobile Information Device Profile (MIDP) Reference Implementation

Mobile Information Device Profile (MIDP) Reference Implementation			
Package/Subcategory	**Error**	**Non-Error**	**Total**
classes	23	1	24
other	174	9	183
doc	32	4	36
ui_doc	4	1	5
jam	86	8	94
ui_tests	10	0	10
spec	242	37	279
rms	42	0	42
networking	216	10	226
demo	52	10	62
emulator	34	3	37
i18n	25	0	25
ui	668	13	681
security	18	5	23
games	51	0	51
usability	36	4	40
tests	33	2	35
testbeans	5	0	5
Total	1751	107	1858

Table A.95: Error and Non-Error Changes for Mobile Information Device Profile and Optional Package JSRS - Phoneme Feature

Mobile Information Device Profile And Optional Package JSRS - Phoneme Feature			
Package/Subcategory	**Error**	**Non-Error**	**Total**
push	45	4	49
rms	25	4	29
usability	87	19	106
security	27	6	33
demo	7	1	8
appmgmt	113	39	152
networking	87	6	93
ui	643	46	689
graphics	172	8	180
jsrop	523	91	614
other	438	92	530
doc	113	16	129
games	18	0	18
multitasking	106	3	109
autotest	18	1	19
i18n	12	7	19
wma	114	5	119
installer	65	9	74
media	417	12	429
portability	48	10	58
Total	3078	379	3457

Table A.96: Error and Non-Error Changes for Mobile Information Device Profile For Palm OS (MIDP)

Mobile Information Device Profile For Palm OS (MIDP)			
Package/Subcategory	**Error**	**Non-Error**	**Total**
usability	29	2	31
other	138	6	144
rms	5	0	5
networking	46	0	46
jam	8	0	8
docs	4	0	4
https	1	1	2
ui	278	14	292
ota	1	0	1
tests	11	1	12
Total	521	24	545

Table A.97: Error and Non-Error Changes for Open ESB - All Categories

Open ESB - All Categories			
Package/Subcategory	**Error**	**Non-Error**	**Total**
documentation	2	1	3
installer	8	0	8
other	0	1	1
visualjava_editor	2	0	2
encoders	0	1	1
bpel	9	0	9
integration_test_driver	15	5	20
Total	36	8	44

Table A.98: Error and Non-Error Changes for Open ESB - Binding Components

Open ESB - Binding Components			
Package/Subcategory	**Error**	**Non-Error**	**Total**
mq	5	0	5
ftp	7	6	13
jms	7	2	9
smtp	23	0	23
jdbc	22	6	28
sap	1	0	1
httpsoap	139	9	148
file	21	4	25
Total	225	27	252

Table A.99: Error and Non-Error Changes for Open ESB - Service Engines

Open ESB - Service Engines			
Package/Subcategory	**Error**	**Non-Error**	**Total**
bpel_se	312	36	348
xslt_se	14	0	14
iep_se	33	0	33
sql_se	25	2	27
Total	384	38	422

Table A.100: Error and Non-Error Changes for Openinstaller – Assembly

Openinstaller - Assembly			
Package/Subcategory	**Error**	**Non-Error**	**Total**
other	3	2	5
Total	3	2	5

Table A.101: Error and Non-Error Changes for Open ESB- Tools

Open ESB- Tools			
Package/Subcategory	**Error**	**Non-Error**	**Total**
iep_project	12	0	12
etl_editor	33	3	36
serviceassembly_editor	6	0	6
iep_editor	8	1	9
etl_project	9	0	9
Total	68	4	72

Table A.102: Error and Non-Error Changes for Openinstaller - Dev Tools

Openinstaller - Dev Tools			
Package/Subcategory	**Error**	**Non-Error**	**Total**
other	4	5	9
ui	7	2	9
assembly	0	1	1
Total	11	8	19

Table A.103: Error and Non-Error Changes for Openinstaller - Quality

Openinstaller - Quality			
Package/Subcategory	**Error**	**Non-Error**	**Total**
unit_tests	28	1	29
other	21	2	23
Total	49	3	52

Table A.104: Error and Non-Error Changes for Openinstaller - Runtime

Openinstaller - Runtime			
Package/Subcategory	**Error**	**Non-Error**	**Total**
install	173	54	227
logging	12	4	16
configuration	19	34	53
other	41	27	68
solaris	6	1	7
uninstall	30	2	32
windows	13	9	22
linux	12	0	12
upgrade	14	2	16
Total	320	133	453

Table A.105: Error and Non-Error Changes for Openinstaller - Scripts

Openinstaller - Scripts			
Package/Subcategory	**Error**	**Non-Error**	**Total**
launcher	12	0	12
uninstaller	3	0	3
other	15	1	16
installer	5	0	5
Total	35	1	36

Table A.106: Error and Non-Error Changes for Openinstaller - User Interface

Openinstaller - User Interface			
Package/Subcategory	**Error**	**Non-Error**	**Total**
other	15	6	21
g11n	0	1	1
l10n	11	0	11
graphical	69	17	86
i18n	14	0	14
textual	35	5	40
silent	0	1	1
usability	31	13	44
Total	175	43	218

Table A.107: Error and Non-Error Changes for for Openinstaller -Docs

Openinstaller -Docs			
Package/Subcategory	**Error**	**Non-Error**	**Total**
docs	1	0	1
other	2	0	2
Total	3	0	3

Table A.108: Error and Non-Error Changes for Openinstaller -Mi5

Openinstaller -Mi5			
Package/Subcategory	**Error**	**Non-Error**	**Total**
mi5	254	21	275
Total	254	21	275

Table A.109: Error and Non-Error Changes for Personal And Embedded Java

Personal And Embedded Java			
Package/Subcategory	**Error**	**Non-Error**	**Total**
platform_tools	22	5	27
classes_util	19	2	21
native_interface	8	2	10
i18n	10	1	11
window_system	9	0	9
doc	39	6	45
classes_security	12	0	12
classes_io	10	3	13
classes_net	12	3	15
javacheck	98	4	102
e_specification	2	0	2
p_specification	2	2	4
classes_awt	110	15	125
graphics_system	43	2	45
classes_lang	26	2	28
runtime	176	28	204
peers	271	3	274
tools	23	4	27
l10n	1	0	1
other	90	14	104
test_tools	43	11	54
Total	1026	107	1133

Table A.110: Error and Non-Error Changes for Portal Server - Mobile Access

Portal Server - Mobile Access			
Package/Subcategory	**Error**	**Non-Error**	**Total**
jcapi	1	0	1
sso	10	8	18
contentprovider_api	1	0	1
mail	245	40	285
admin	51	2	53
pab_ui	63	6	69
exchange	3	1	4
usability	3	3	6
lotus_notes	2	0	2
performance	4	0	4
x86	3	0	3
jabapi	1	0	1
bea-wl	4	1	5
ibm-ws	1	0	1
windows	4	0	4
linux	4	0	4
taglib	74	11	85
l10n	35	2	37
mail_ui	185	21	206
session	1	0	1
calendar_ui	248	19	267
desktop	473	60	533
clientdetect	30	5	35
install	116	6	122
calendar	227	22	249
aligo	111	15	126
other	145	28	173
i18n	73	0	73
pab	81	8	89
l10n-asia	37	0	37
authentication	90	6	96
voice	46	9	55
doc	92	17	109
Total	2464	290	2754

Table A.111: Error and Non-Error Changes for Openinstaller -Release Engineering

Openinstaller -Release Engineering			
Package/Subcategory	**Error**	**Non-Error**	**Total**
tools	1	4	5
other	8	3	11
builds	24	4	28
Total	33	11	44

Table A.112: Error and Non-Error Changes for Portal Server - Search

Portal Server - Search			
Package/Subcategory	**Error**	**Non-Error**	**Total**
taxonomy	94	5	99
other	84	11	95
database	274	28	302
admin	449	53	502
robot	218	33	251
ui	146	26	172
pks	223	6	229
doc	36	2	38
migration	29	2	31
install	56	4	60
usability	10	0	10
javasdk	9	1	10
performance	32	6	38
libnet	1	0	1
security	4	0	4
searchengine	165	19	184
build	4	4	8
l10n	13	0	13
import	54	5	59
i18n	42	3	45
server	48	10	58
Total	1991	218	2209

Table A.113: Error and Non-Error Changes for Portal Server - Secure Remote Access

Portal Server - Secure Remote Access			
Package/Subcategory	**Error**	**Non-Error**	**Total**
sun-iws	4	1	5
bea-wl	31	1	32
windows	9	2	11
proxylet	351	48	399
install	239	21	260
data_migration	94	2	96
netlet	533	131	664
erproxy	954	215	1169
netfile	1013	135	1148
doc	176	15	191
admin	179	14	193
sun-as	29	2	31
ibm-ws	13	0	13
l10n	40	0	40
Total	3665	587	4252

Table A.114: Error and Non-Error Changes for Service Registry

Service Registry			
Package/Subcategory	**Error**	**Non-Error**	**Total**
jaxr	71	3	74
server	144	5	149
install	108	10	118
admin	25	3	28
l10n	63	1	64
i18n	23	3	26
web-ui	418	37	455
java-ui	18	0	18
jaxr-ra	10	0	10
uddi	1	0	1
doc	20	5	25
Total	901	67	968

Table A.115: Error and Non-Error Changes for Scripting

Scripting			
Package/Subcategory	**Error**	**Non-Error**	**Total**
tools	4	2	6
javascript	29	4	33
api	24	5	29
Total	57	11	68

Table A.116: Error and Non-Error Changes for Soap With Attachments Api For Java

Soap With Attachments Api For Java			
Package/Subcategory	**Error**	**Non-Error**	**Total**
docs	12	0	12
soap1.1	113	10	123
spec	7	5	12
soap1.2	46	2	48
Total	178	17	195

Table A.117: Error and Non-Error Changes for Standard Javadoc Doclet

Standard Javadoc Doclet			
Package/Subcategory	**Error**	**Non-Error**	**Total**
tbd	1011	373	1384
stddoclet	41	11	52
makefile	10	0	10
tagletspec	0	3	3
	1062	387	1449
Total	1011	373	1384

Table A.118: Error and Non-Error Changes for Sun Java Studio

Sun Java Studio			
Package/Subcategory	**Error**	**Non-Error**	**Total**
vcs-generic	349	38	387
examples	307	19	326
core	353	33	386
jdbc-client	172	11	183
repository	3	1	4
startup	91	6	97
other	116	8	124
editor	89	19	108
form	36	5	41
tpruntime	314	26	340
refactoring	257	17	274
jarpackager	36	0	36
j2ee-assembly	569	27	596
vcscore	23	1	24
dbschema	164	13	177
docs	218	12	230
utilities	18	1	19
tpui	536	44	580
jsp	1523	114	1637
fastjavac	304	10	314
xml	269	14	283
ejb-assembly	613	23	636
ri	260	17	277
webservices-	1355	79	1434
installer	668	69	737
application-client	147	6	153
nct	3	1	4
ejb_workshop	964	78	1042
dbexplorer	104	14	118
kjava	595	97	692
ifdef	24	2	26
sim	95	4	99
golive	22	0	22
filecopy	2	0	2
debuggercore	30	0	30
pointbase	93	6	99
rmi	35	2	37
corba	210	19	229
server-api	184	17	201
projects	18	6	24
javacvs	7	0	7

Table A.118 (cont'd…)

webservices-testrun	21	4	25
ias	164	4	168
jdbc-drivers	26	1	27
javadoc	5	2	7
webservices-execution	100	3	103
vcscvs	20	3	23
xml-client	56	4	60
java	33	1	34
jndi	6	0	6
objectbrowser	3	0	3
dreamweaver	27	0	27
properties	6	0	6
vcs-teamware	5	2	7
i18n_support	5	0	5
appserver	131	21	152
ejb-testrun	62	11	73
xml-editor	2	2	4
blitz	21	0	21
me_installer	43	0	43
vss_scc	40	2	42
welcome_screen	4	1	5
crashreporter	10	0	10
appframework	6	0	6
updatetool	12	0	12
pvcs_scc	62	7	69
jatox-components	1	0	1
tasklist	1	0	1
image	0	1	1
j2eedebugger	73	4	77
debuggerjpda	9	1	10
webserver	23	7	30
debuggertools	10	0	10
apisupport	4	0	4
ant	9	2	11
clazz	1	0	1
beans	4	0	4
importear	149	12	161
cobundle-installer	162	16	178
performance-tests	2	0	2
junit	1	0	1
editor	2707	426	3133
clazz	66	6	72
usersguide	270	31	301

Table A.118 (cont'd…)

httpserver	95	11	106
objectbrowser	198	13	211
text	38	8	46
autoupdate	472	66	538
i18n	161	25	186
image	27	6	33
apisupport	127	50	177
openidex	10	2	12
icebrowser	97	3	100
applet	49	5	54
ui	389	97	486
beans	194	37	231
javadoc	469	67	536
other	1	1	2
projects	1421	147	1568
core	7633	554	8187
vcscvs	231	36	267
java	2798	328	3126
jarpackager	436	33	469
openide	106	8	114
form	1715	157	1872
openide	4233	549	4782
utilities	338	56	394
debuggerjpda	461	41	502
web	2398	223	2621
debuggercore	1526	227	1753
debuggertools	49	1	50
libs	19	2	21
classfile	28	4	32
db	265	31	296
installer	356	38	394
cpp	64	13	77
monitor	136	34	170
externaleditor	80	8	88
junit	167	40	207
sim	112	17	129
mdr	189	33	222
filecopy	11	1	12
schema2beans	46	7	53
scripting	44	5	49
classclosure	3	0	3
extbrowser	127	18	145
tomcatint	446	34	480

Table A.118 (cont'd…)

properties	192	21	213
html	33	8	41
vcscore	766	83	849
nbbuild	202	60	262
javacvs	601	76	677
rmi	125	10	135
jndi	57	2	59
diff	86	17	103
corba	123	12	135
ant	580	96	676
j2eeserver	310	11	321
logger	14	4	18
xml	537	109	646
vcsgeneric	1037	117	1154
Total	47966	5095	53061

Table A.119: Error and Non-Error Changes for Sun Studio – Analyzer

Sun Studio - Analyzer			
Package/Subcategory	**Error**	**Non-Error**	**Total**
libcollector	2	0	2
other	1	2	3
performance	17	1	18
mpi	281	78	359
Total	301	81	382

Table A.120: Error and Non-Error Changes for Sun Studio - C Compiler

Sun Studio - C Compiler			
Package/Subcategory	**Error**	**Non-Error**	**Total**
indent	1	0	1
lint	4	0	4
driver	6	2	8
stabs	3	0	3
gcc-compat	2	2	4
compiler	45	8	53
Total	61	12	73

Table A.121: Error and Non-Error Changes for Sun Java System Portalserver

Sun Java System Portalserver			
Package/Subcategory	**Error**	**Non-Error**	**Total**
config	107	14	121
naming	2	0	2
sun-iws	13	2	15
pll	9	2	11
session	36	16	52
firewall	5	2	7
patch	94	4	98
portlets	711	109	820
admincli	293	22	315
community	300	89	389
mab	34	1	35
federated_identity	7	0	7
authentication	190	75	265
wss	117	11	128
cms	137	14	151
workflow	75	1	76
docs-windows	2	0	2
hp-ux	64	1	65
admintaglib	82	8	90
tck	15	0	15
fatwire	2	0	2
windows	159	3	162
jsfportlet	26	5	31
adminconsole	636	59	695
tools	136	6	142
desktop	2054	423	2477
i18n	234	9	243
profile	111	30	141
file	39	10	49
other	523	115	638
logging	108	24	132
admin	836	152	988
doc	692	129	821
rewriter	521	86	607
install	1893	159	2052
contentprovider	215	63	278
netmail	373	131	504
compass	29	6	35
l10n-asia	49	0	49
wsrp	464	27	491
l10n	594	7	601

Table A.121 (cont'd…)

ibm-ws	48	1	49
sampleportal	648	68	716
portletcontainer	291	26	317
sun-as	86	1	87
bea-wl	131	9	140
yahoo	172	30	202
migration	853	25	878
Total	14216	1975	16191

Table A.122: Error and Non-Error Changes for Sun Studio - C++ Compiler

Sun Studio - C++ Compiler			
Package/Subcategory	**Error**	**Non-Error**	**Total**
other	1	0	1
compiler	84	7	91
library	40	1	41
generate	4	1	5
template	79	2	81
driver	2	0	2
preproc	7	2	9
	217	13	230
Total	1	0	1

Table A.123: Error and Non-Error Changes for Sun Studio - Distributed Make

Sun Studio - Distributed Make			
Package/Subcategory	**Error**	**Non-Error**	**Total**
other	1	0	1
dmake	0	1	1
Total	1	1	2

Table A.124: Error and Non-Error Changes for Sun Studio - C/C++/Fortran Compilers And Tools – Misc

Sun Studio - C/C++/Fortran Compilers And Tools – Misc			
Package/Subcategory	**Error**	**Non-Error**	**Total**
libld_annotate	8	0	8
assembler-x86	11	4	15
libdwarf	12	5	17
x86-performance	115	86	201
libmopt	2	0	2
libdbggen	12	1	13
optimizer	41	2	43
driver	3	2	5
ube	82	4	86
fast-cgen-x86	24	0	24
tcov	0	1	1
cgen	11	2	13
postopt	2	0	2
install	1	0	1
profiling	1	0	1
libdisasm-x86	1	0	1
ipo	1	0	1
assembler	4	2	6
fast-cgen	2	0	2
fw_egret	4	0	4
other	19	8	27
libmtsk	0	1	1
headers	0	1	1
ld	1	0	1
Total	357	119	476

Table A.125: Error and Non-Error Changes for Sun Studio - Dbx

Sun Studio - Dbx			
Package/Subcategory	**Error**	**Non-Error**	**Total**
commands	3	0	3
c	1	0	1
c++	5	0	5
machine_level	1	0	1
shell	2	1	3
other	19	2	21
runtime_checks	7	0	7
proc_control	3	0	3
mt	1	0	1
dwarf	2	0	2
events	0	1	1
Total	44	4	48

Table A.126: Error and Non-Error Changes Sun Studio - Fortran Compiler

Sun Studio - Fortran Compiler			
Package/Subcategory	**Error**	**Non-Error**	**Total**
compiler	32	1	33
driver	1	1	2
other	5	0	5
gpc	0	1	1
library	8	0	8
middle-end	28	0	28
Total	74	3	77

Table A.127: Error and Non-Error Changes for Sun Studio - Installer

Sun Studio - Installer			
Package/Subcategory	**Error**	**Non-Error**	**Total**
pkg_scripts	1	0	1
install	2	1	3
registration-client	1	0	1
Total	4	1	5

Table A.128: Error and Non-Error Changes for Sun Studio - IDE

Sun Studio - IDE			
Package/Subcategory	**Error**	**Non-Error**	**Total**
glue	19	15	34
d-light	111	18	129
debugger	11	6	17
netbeans	1	1	2
other	3	1	4
startup-problems	1	1	2
base	1	0	1
debugger	1	0	1
other	2	0	2
Total	150	42	192

Table A.129: Error and Non-Error Changes for Sun Studio - Performance Libraries

Sun Studio - Performance Libraries			
Package/Subcategory	**Error**	**Non-Error**	**Total**
install	1	0	1
performance	1	0	1
correctness	2	0	2
Total	4	0	4

APPENDIX B

POWER LAW DISTRIBUTION AND WEIBULL DISTRIBUTION

A. Power Law Distribution

NetBeans

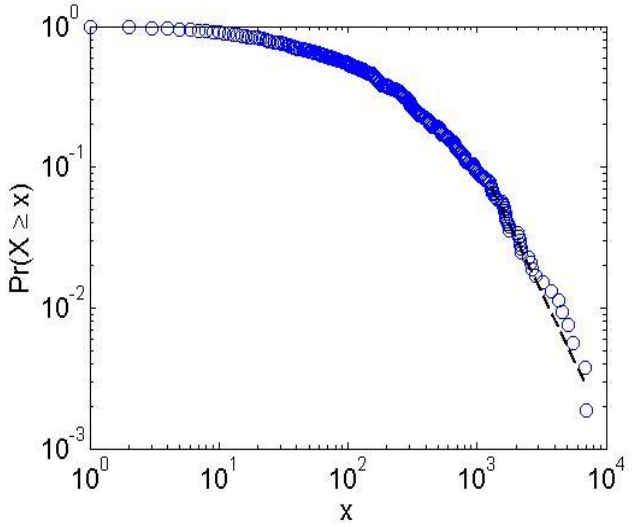

Figure B.1: Plot of Power Law for Total Change in NetBeans

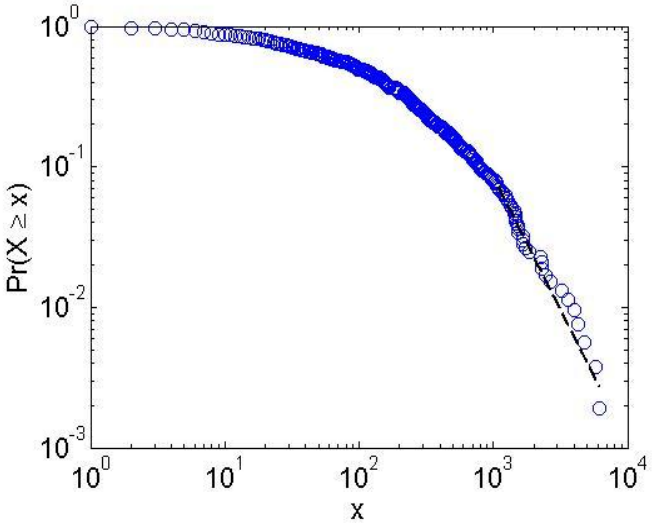

Figure B.2: Plot of Power Law for Error Change in NetBeans

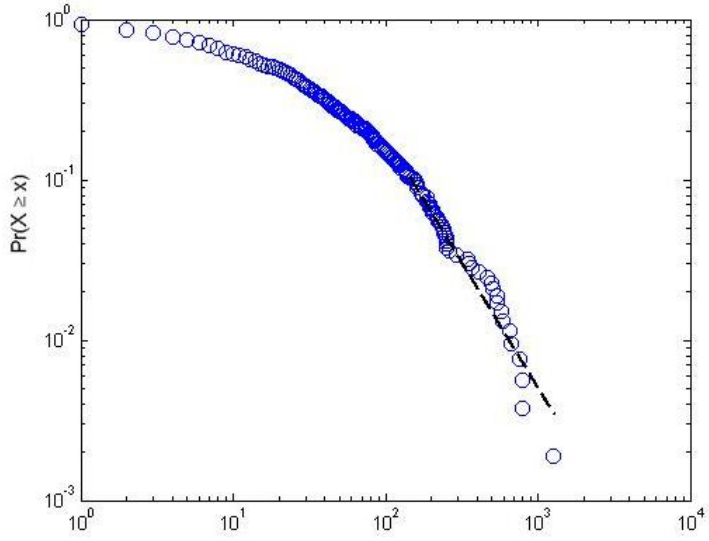

Figure B.3: Plot of Power Law for Non-Error Change in NetBeans

Sun Java Studio

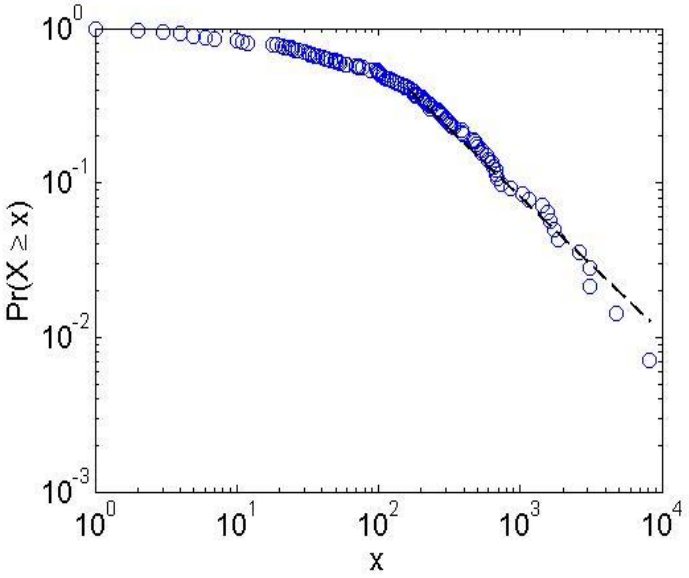

Figure B.4: Plot of Power Law for Total Change in Sun Java Studio

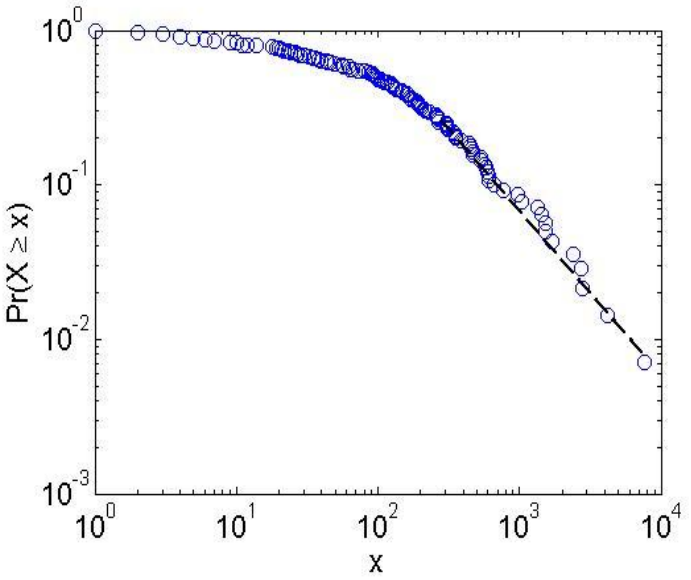

Figure B.5: Plot of Power Law for Error Change in Sun Java Studio

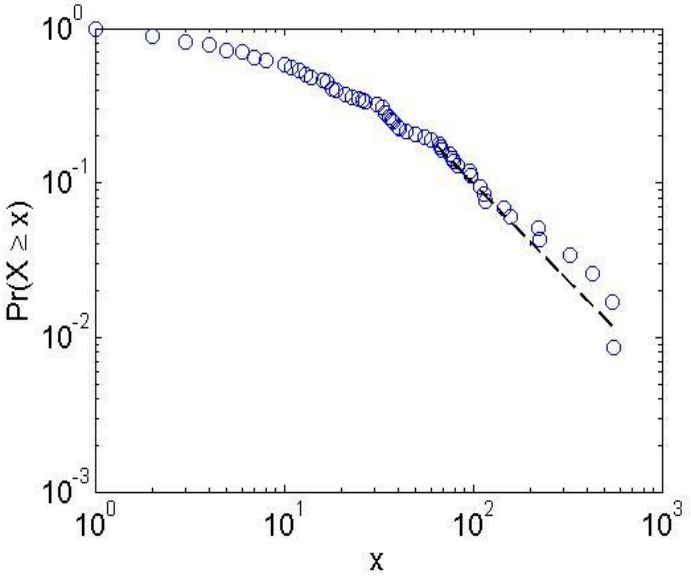

Figure B.6: Plot of Power Law for Non-Error Change in Sun Java Studio

Java SE (JDK/JRE)

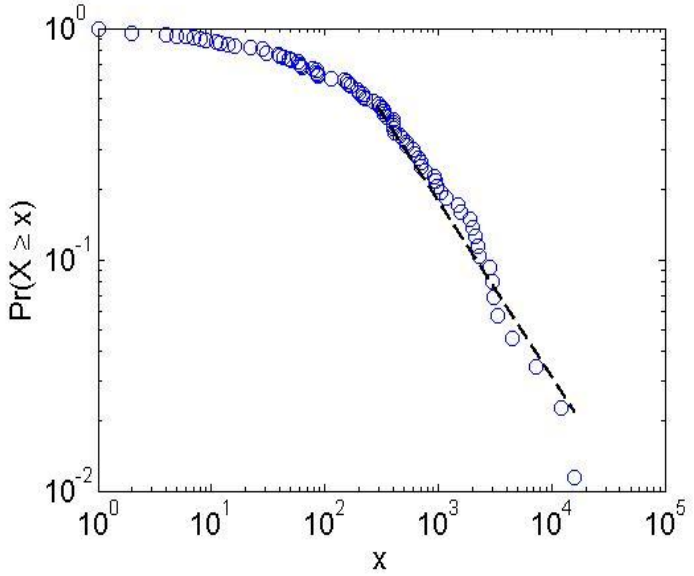

Figure B.7: Plot of Power Law for Total Change in Java SE (JDK/JRE)

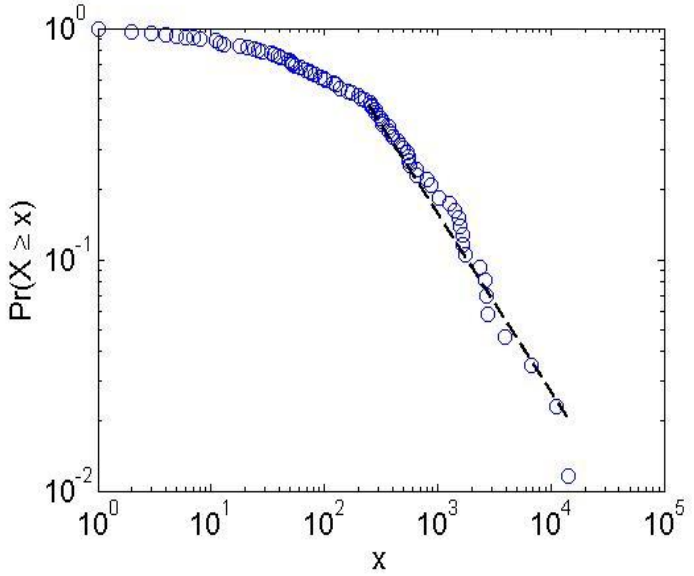

Figure B.8: Plot of Power Law for Error Change in Java SE (JDK/JRE)

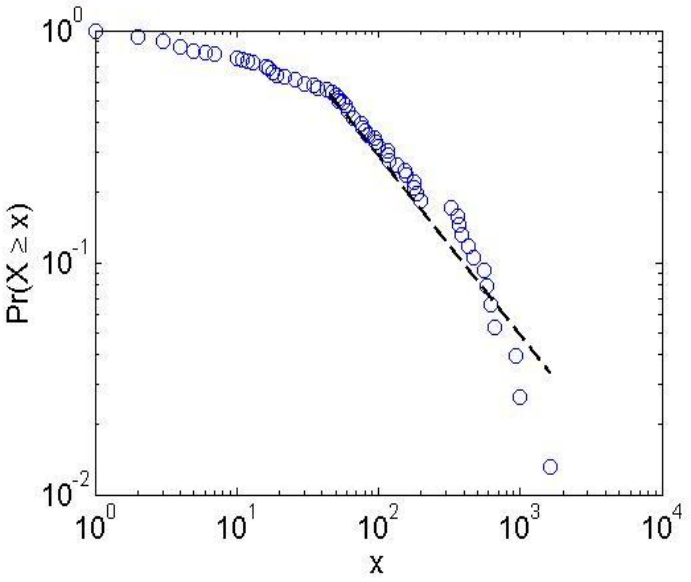

Figure B.9: Plot of Power Law for Non-Error Change in Java SE (JDK/JRE)

Sun Studio

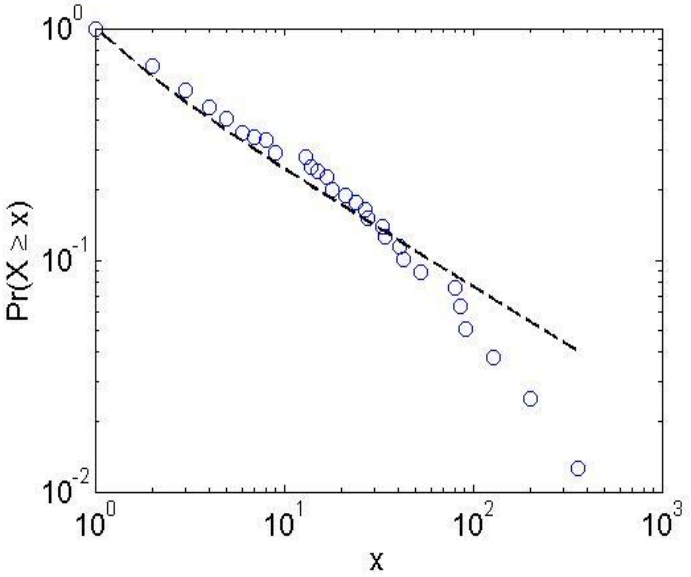

Figure B.10: Plot of Power Law for Total Change in Sun Studio

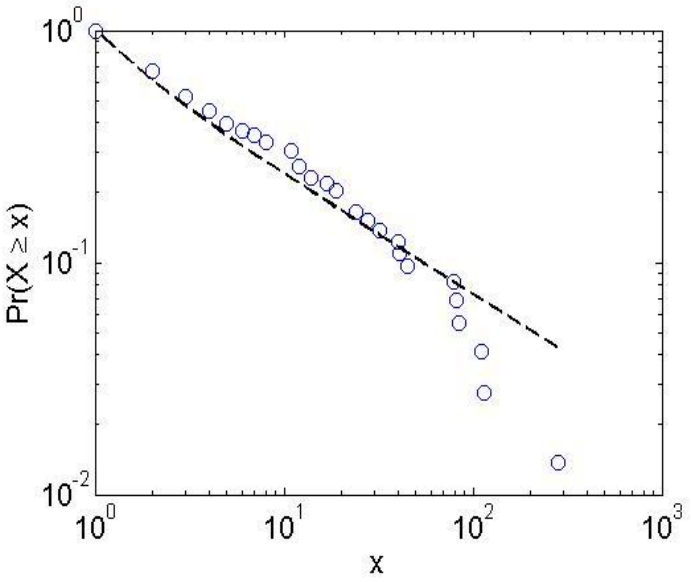

Figure B.11: Plot of Power Law for Error Change in Sun Studio

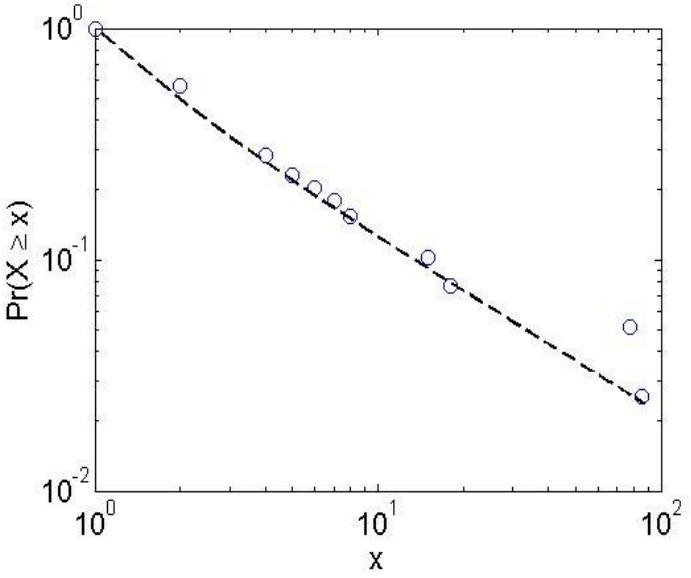

Figure B.12: Plot of Power Law for Non-Error Change in Sun Studio

Jini Network Technology

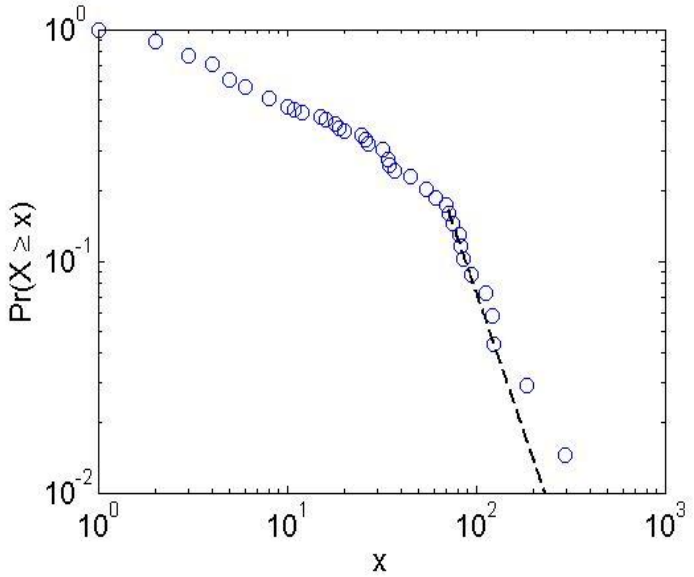

Figure B.13: Plot of Power Law for Total Change in Jini Network Technology

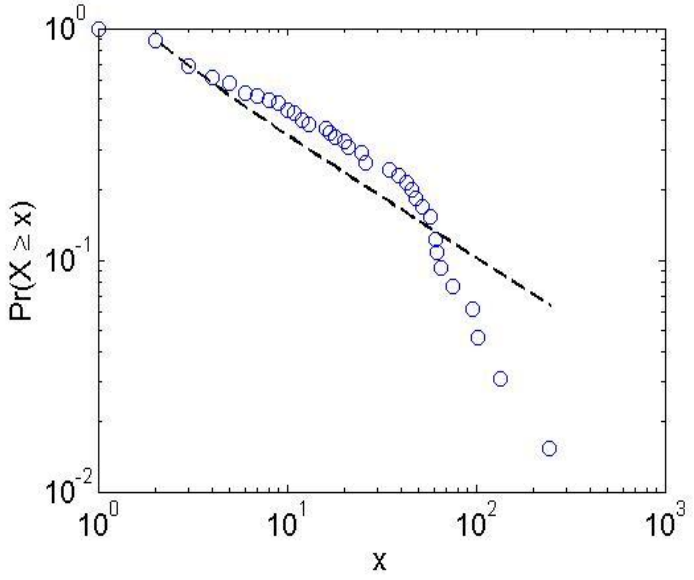

Figure B.14: Plot of Power Law for Error Change in Jini Network Technology

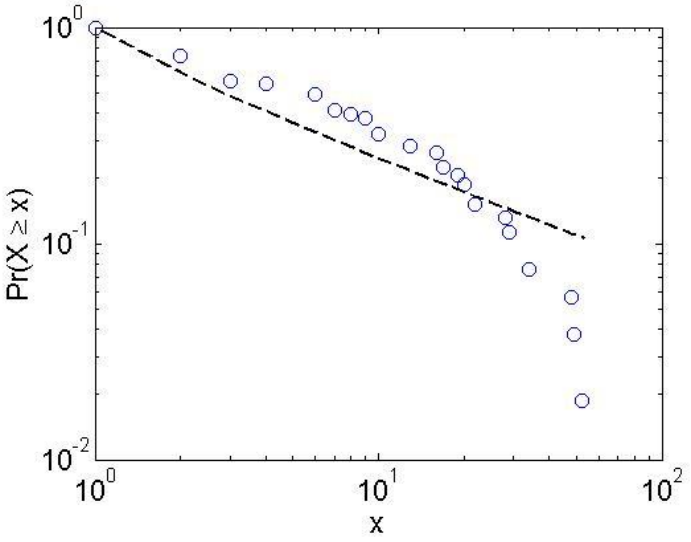

Figure B.15: Plot of Power Law for Non-Error Change in Jini Network Technology

B. Weibull Distribution

NetBeans

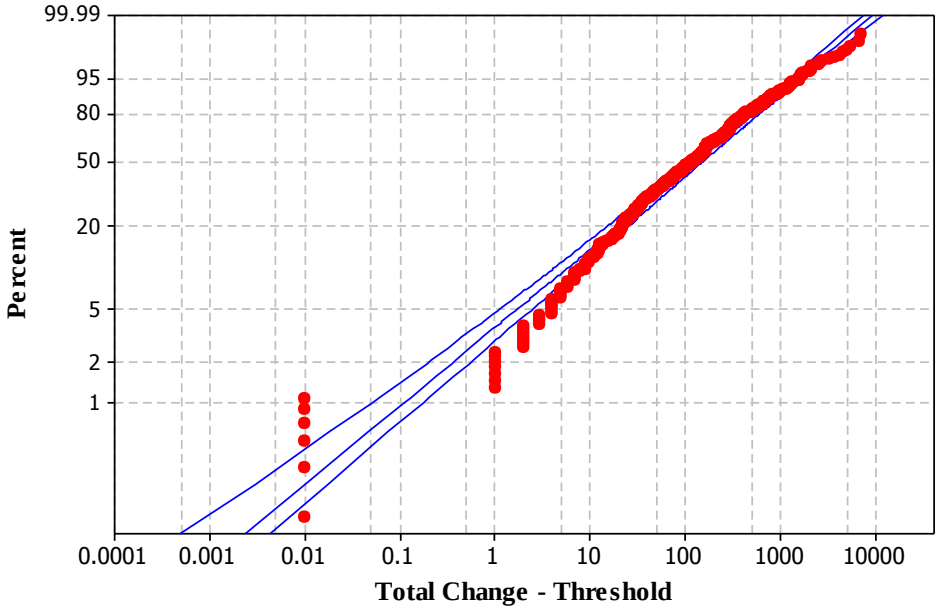

Figure B.16: Plot of Weibull distribution for Total Change in NetBeans

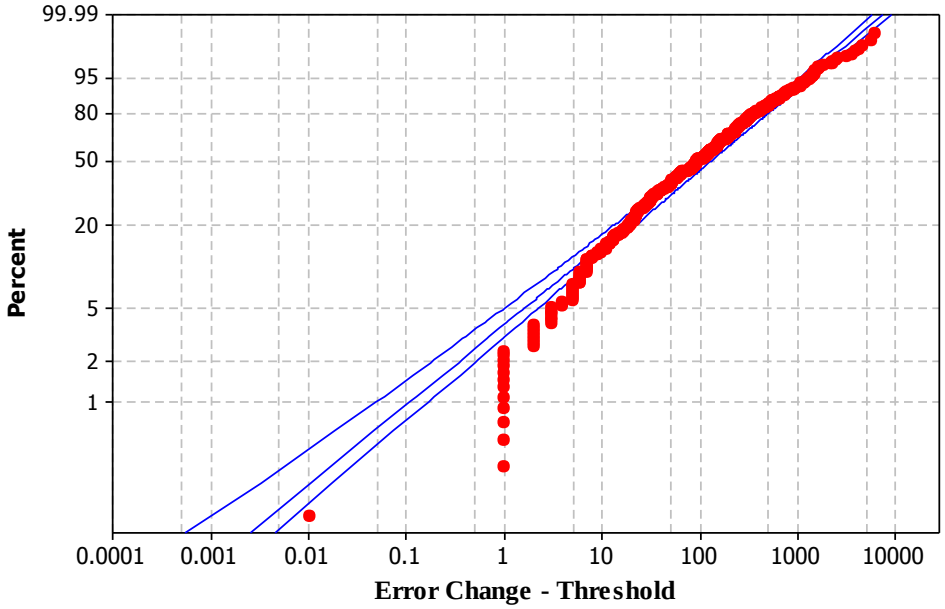

Figure B.17: Plot of Weibull distribution for Error Change in NetBeans

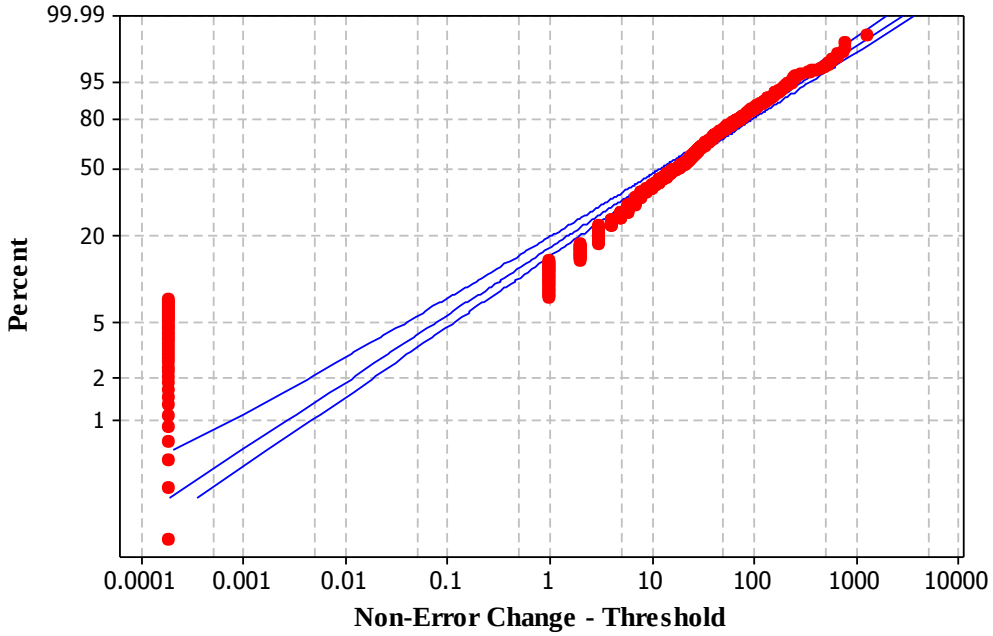

Figure B.18: Plot of Weibull distribution for Non-Error Change in NetBeans

Sun Java Studio

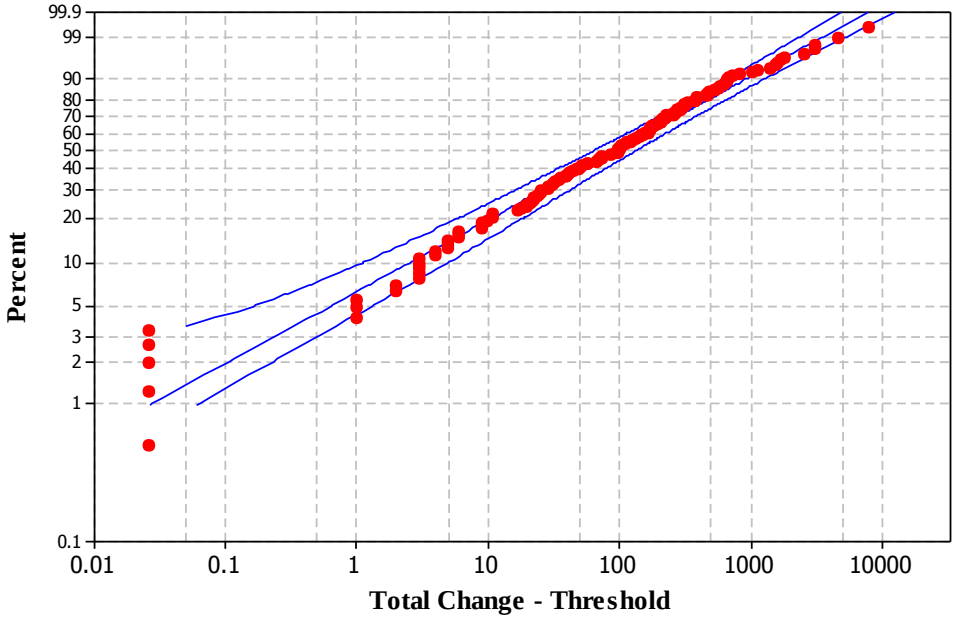

Figure B.19: Plot of Weibull distribution for Total Change in Sun Java Studio

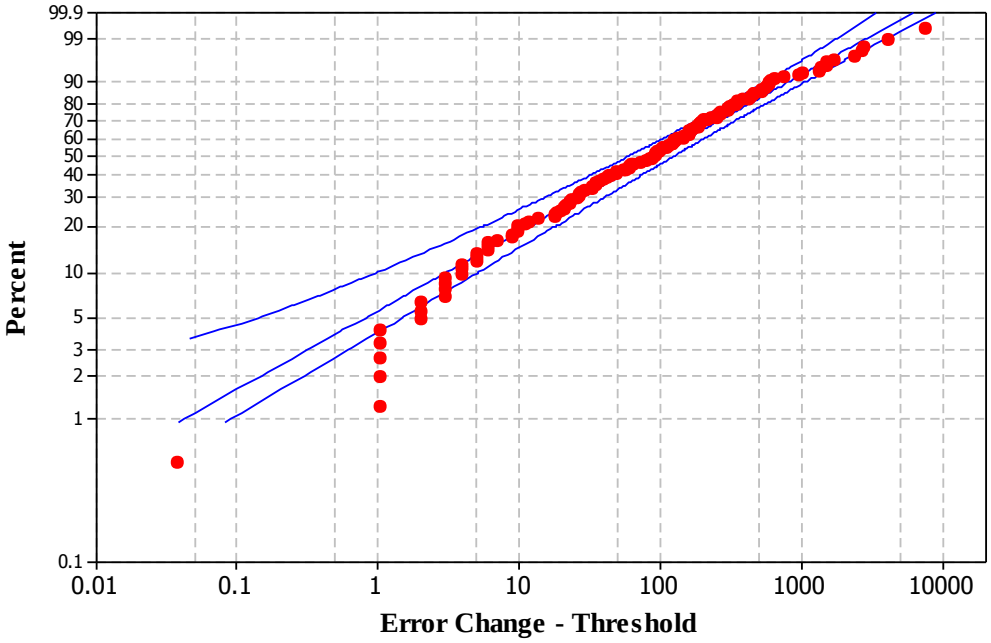

Figure B.20: Plot of Weibull distribution for Error Change in Sun Java Studio

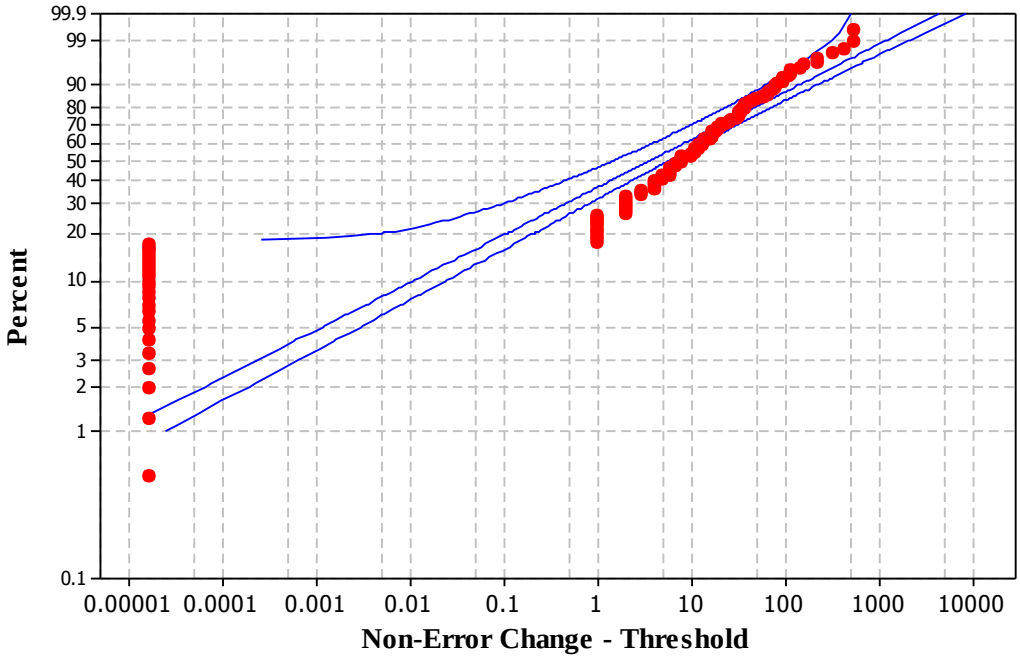

Figure B.21: Plot of Weibull distribution for Non-Error Change in Sun Java Studio

Java SE (JDK/JRE)

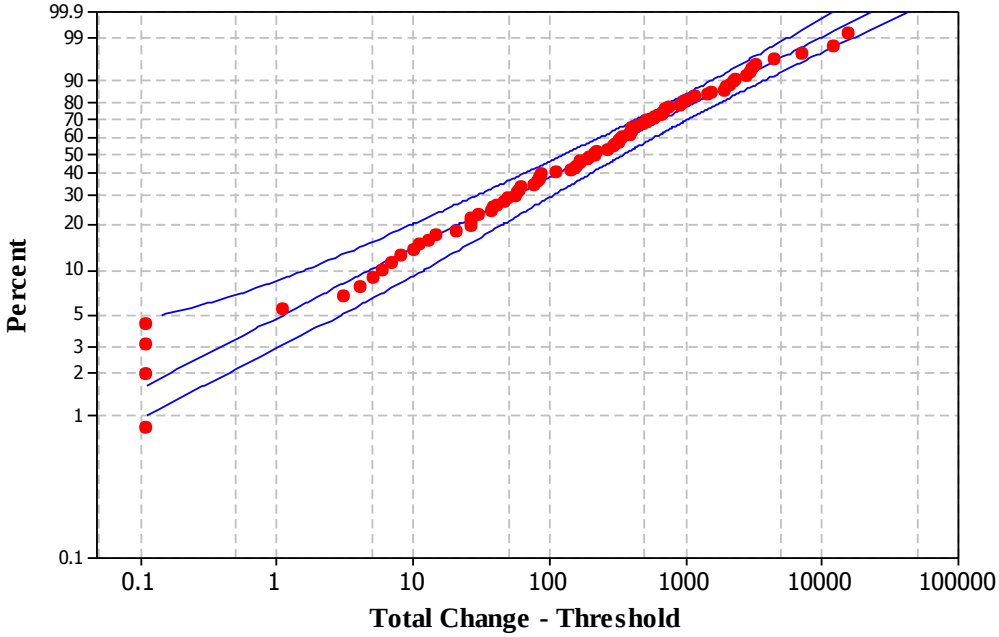

Figure B.22: Plot of Weibull distribution for Total Change in Java SE (JDK/JRE)

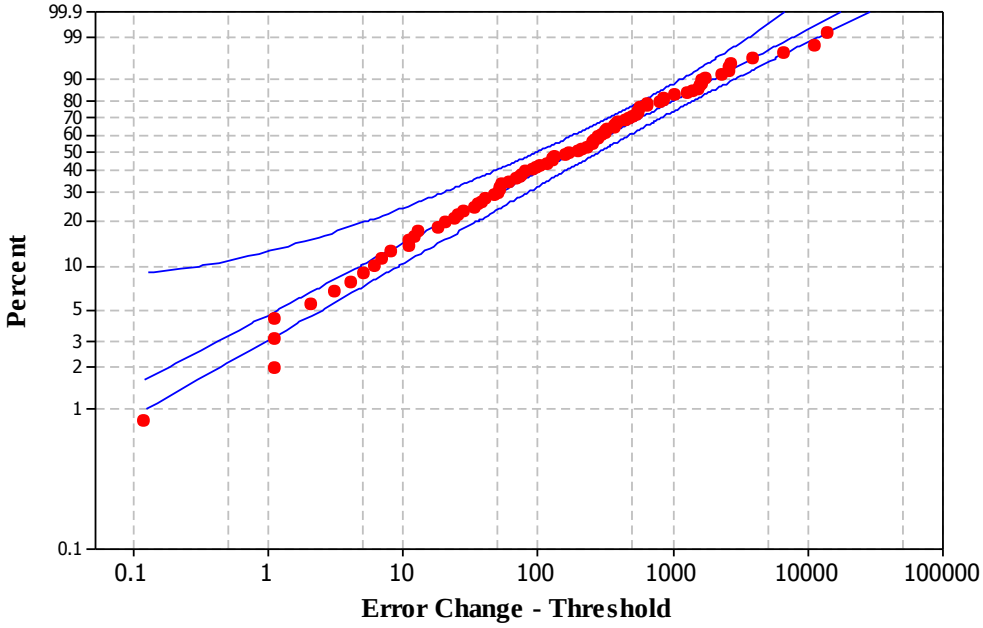

Figure B.23: Plot of Weibull distribution for Error Change in Java SE (JDK/JRE)

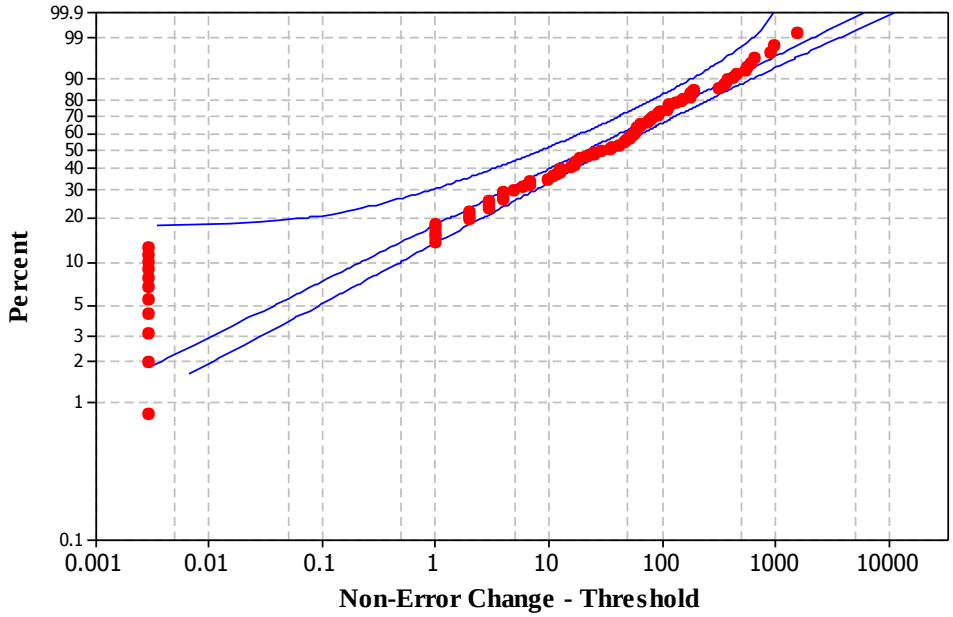

Figure B.24: Plot of Weibull distribution for Non-Error Change in Java SE (JDK/JRE)

Sun Studio

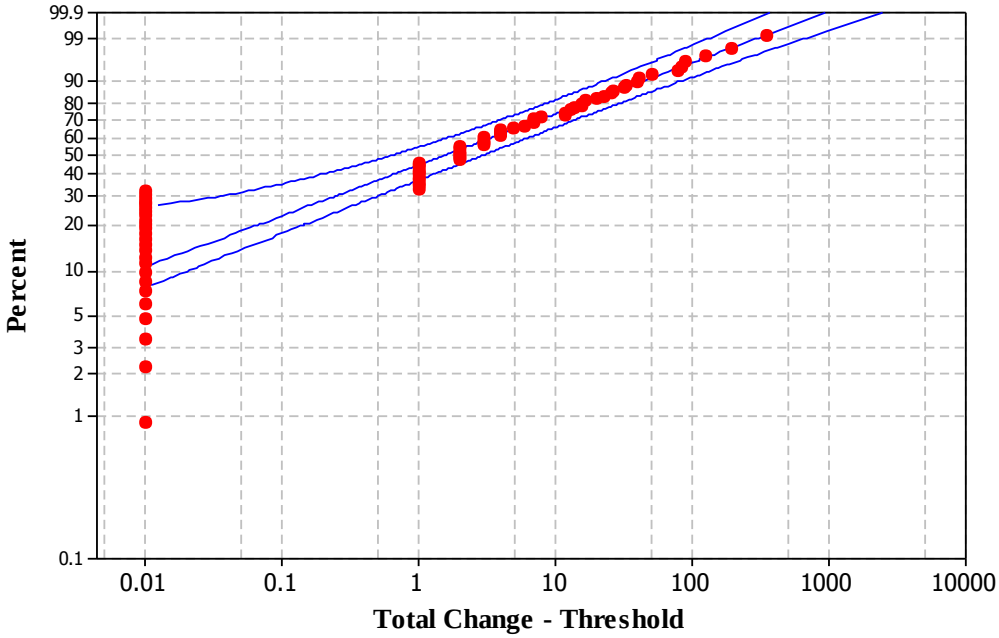

Figure B.25: Plot of Weibull distribution for Total Change in Sun Studio

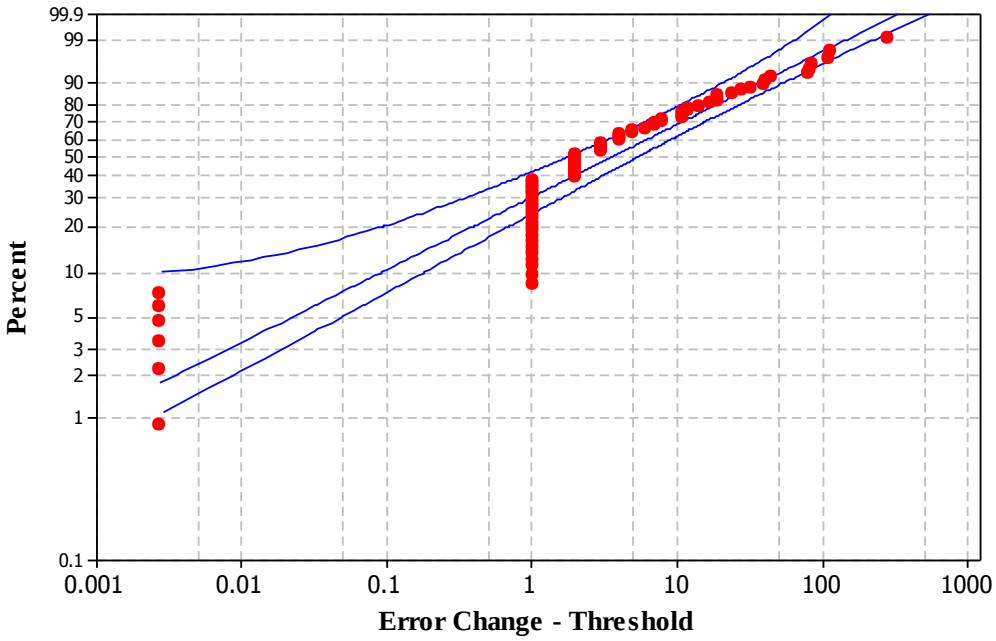

Figure B.26: Plot of Weibull distribution for Error Change in Sun Studio

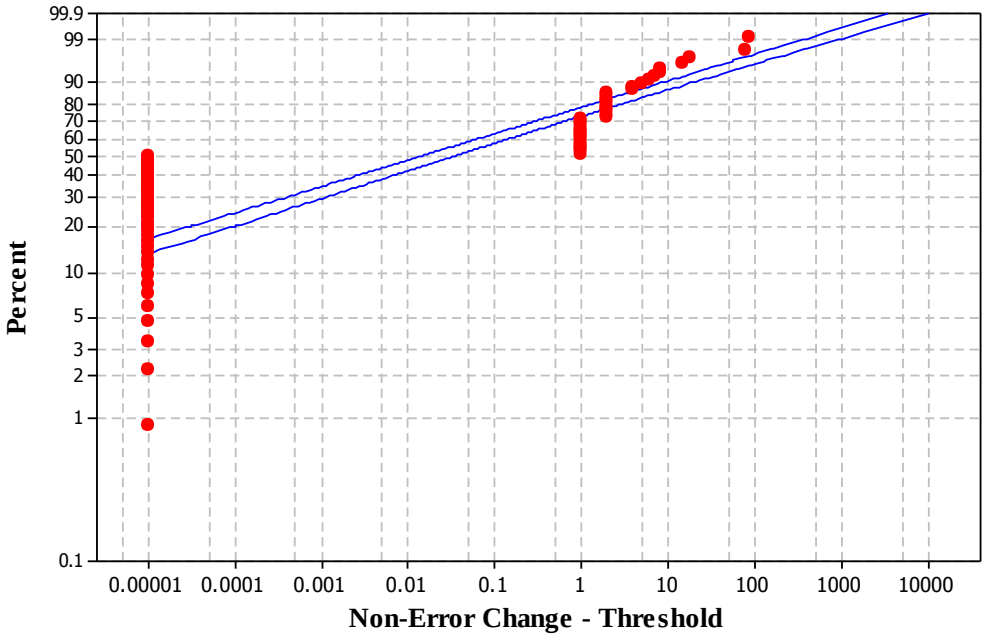

Figure B.27: Plot of Weibull distribution for Non-Error Change in Sun Studio

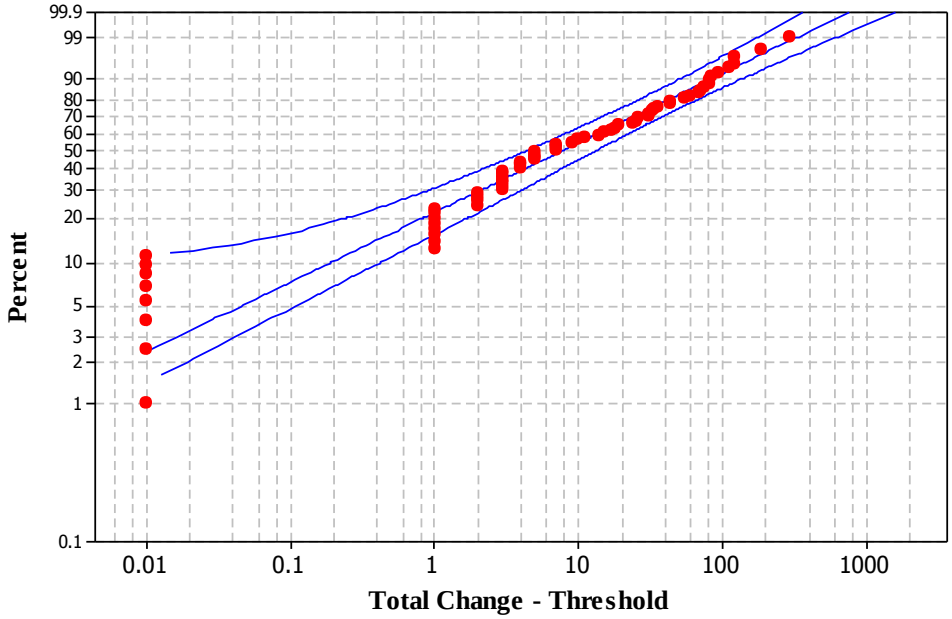

Figure B.28: Plot of Weibull distribution for Total Change in Jini Network Technology

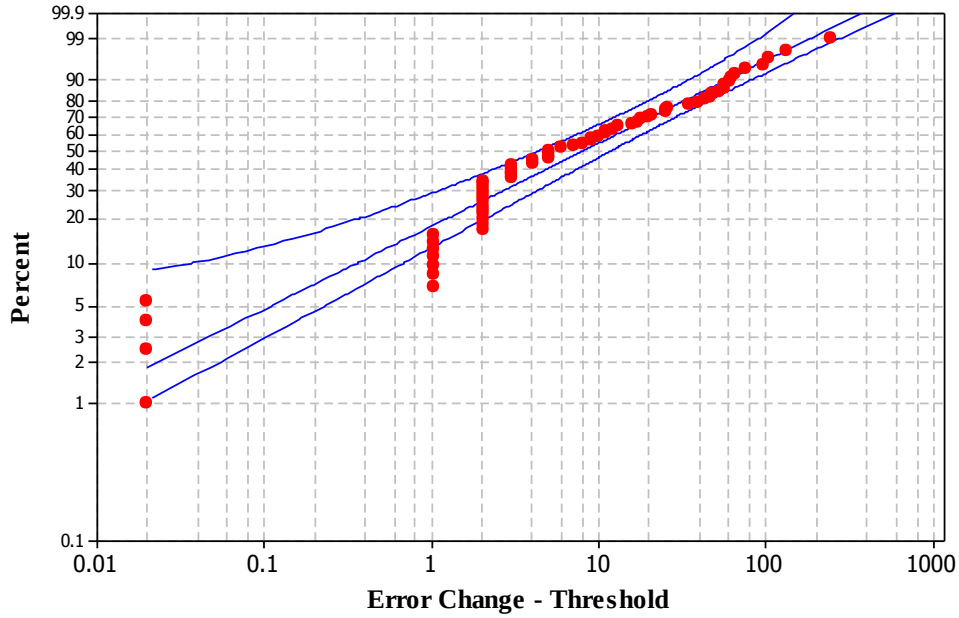

Figure B.29: Plot of Weibull distribution for Error Change in Jini Network Technology

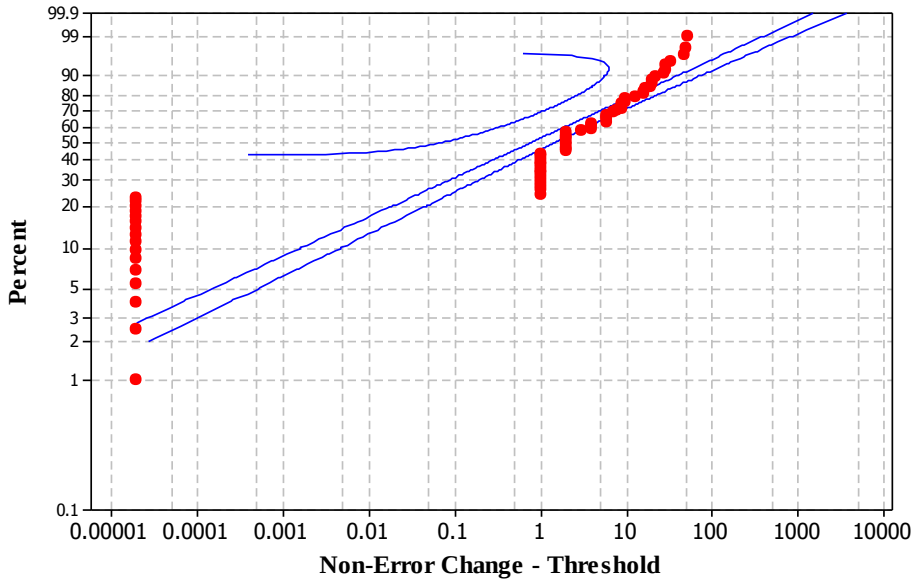

Figure B.30: Plot of Weibull distribution for Non- Error Change in Jini Network Technology

Sun Java System Portal server

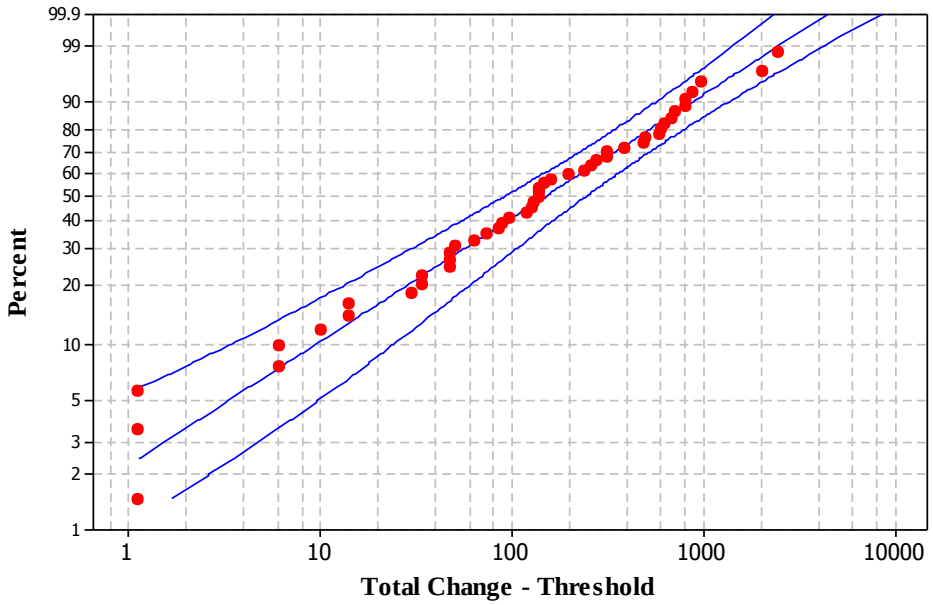

Figure B.31: Plot of Weibull distribution for Total Change in Sun Java System Portal server

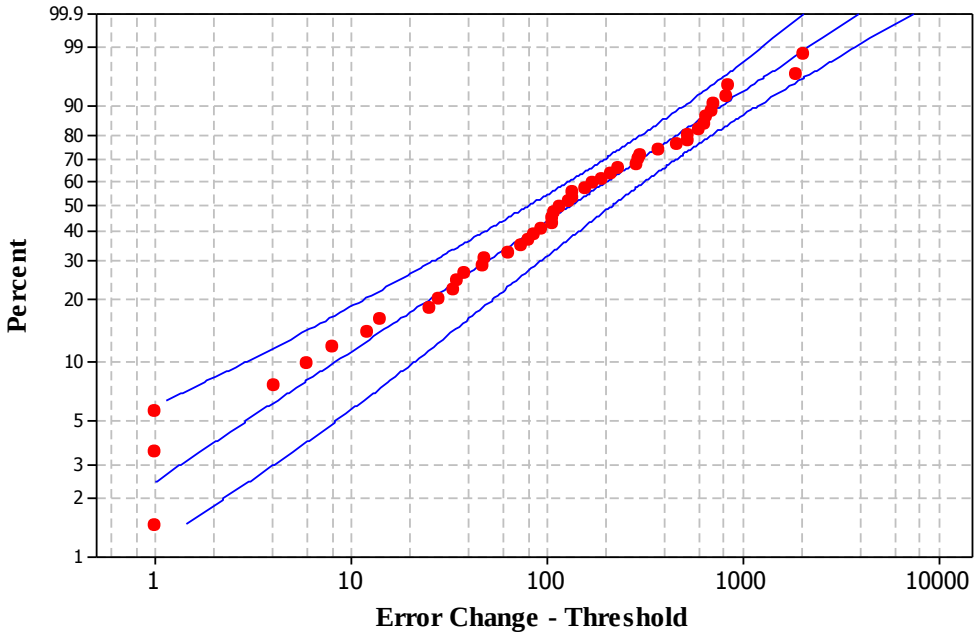

Figure B.32: Plot of Weibull distribution for Error Change in Sun Java System Portal server

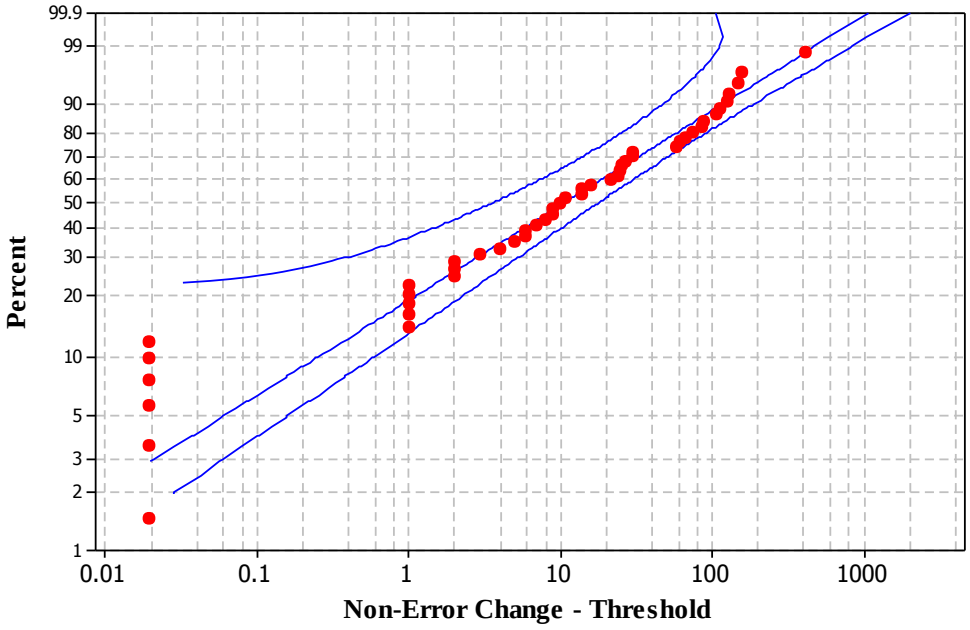

Figure B.33: Plot of Weibull distribution for Non- Error Change in Sun Java System Portal server

Portal Server - Mobile Access

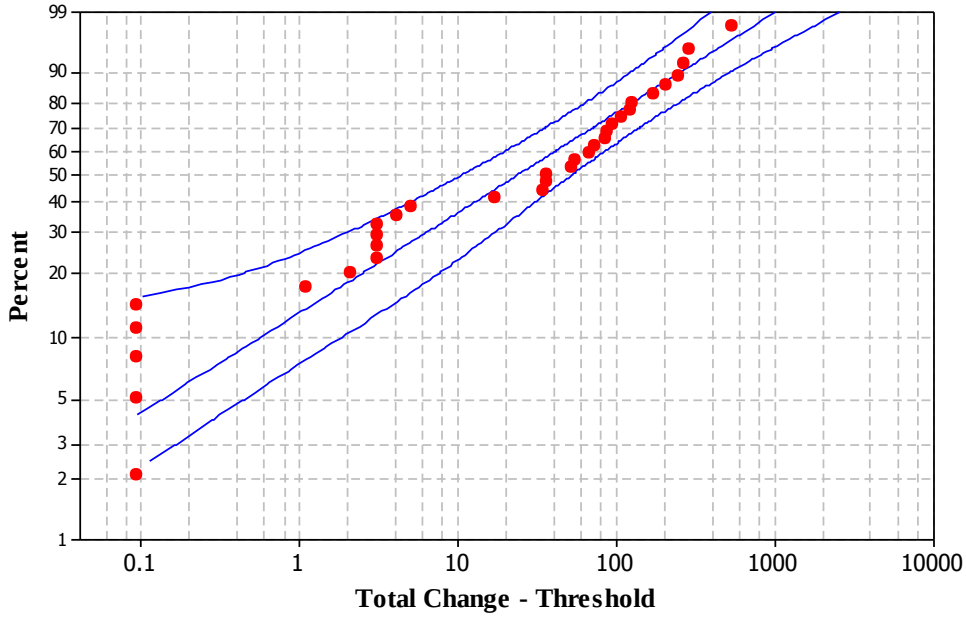

Figure B.34: Plot of Weibull distribution for Total Change in Portal Server - Mobile Access

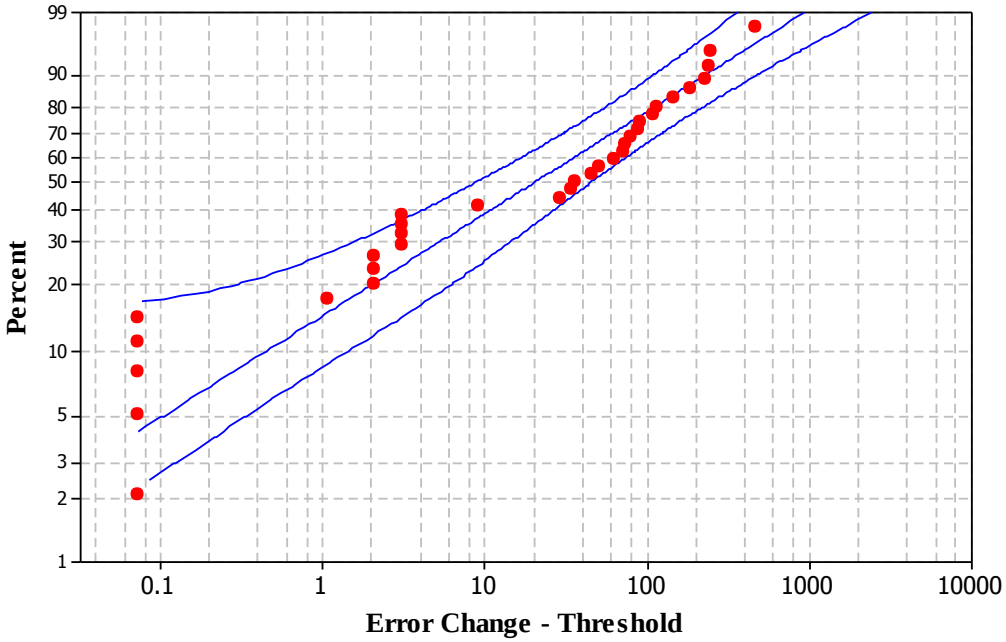

Figure B.35: Plot of Weibull distribution for Error Change in Portal Server - Mobile Access

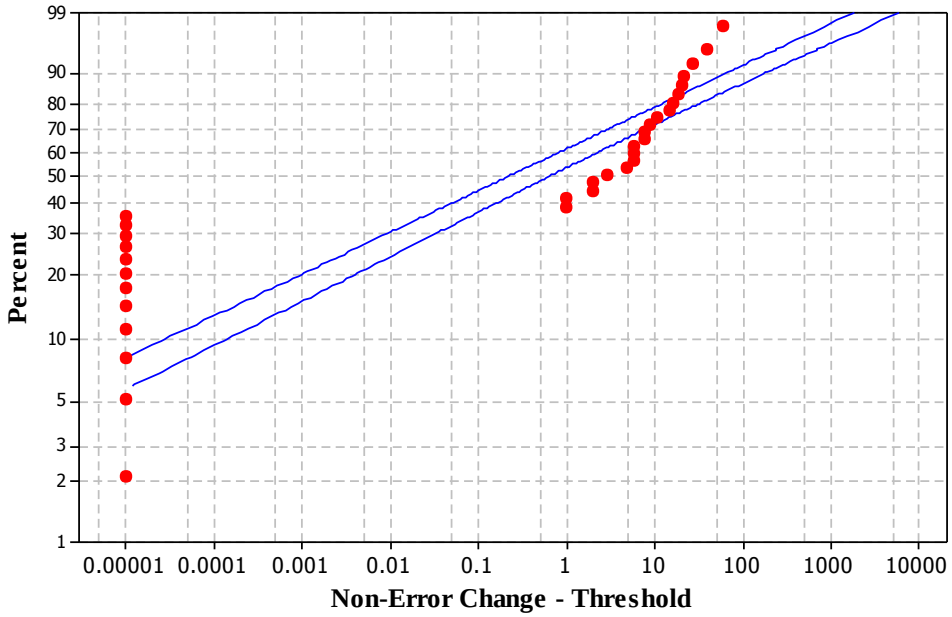

Figure B.36: Plot of Weibull distribution for Non- Error Change in Portal Server Mobile Access

Java EE (Enterprise Edition)

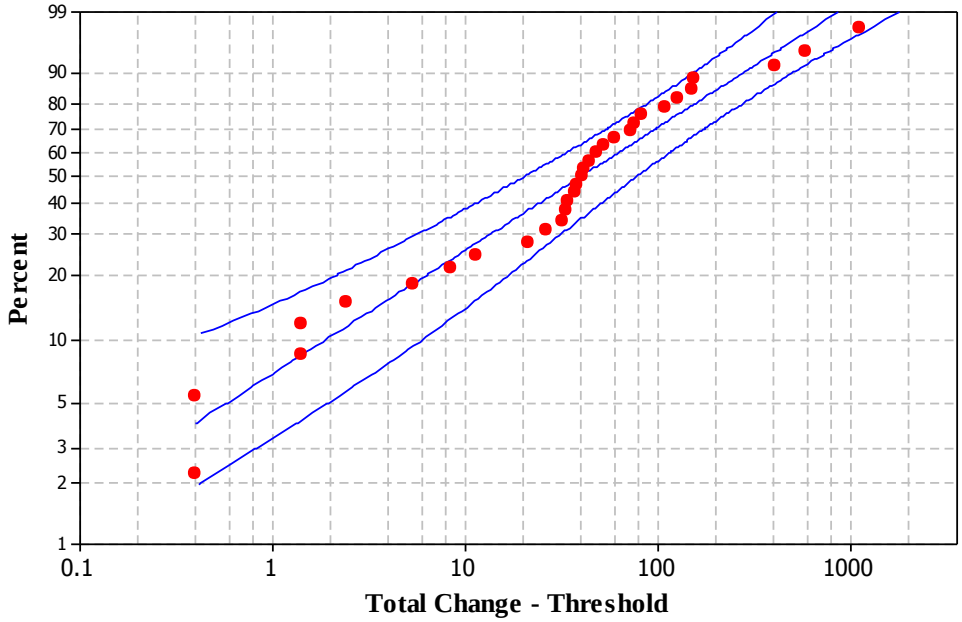

Figure B.37: Plot of Weibull distribution for Total Change in Java Enterprise Edition

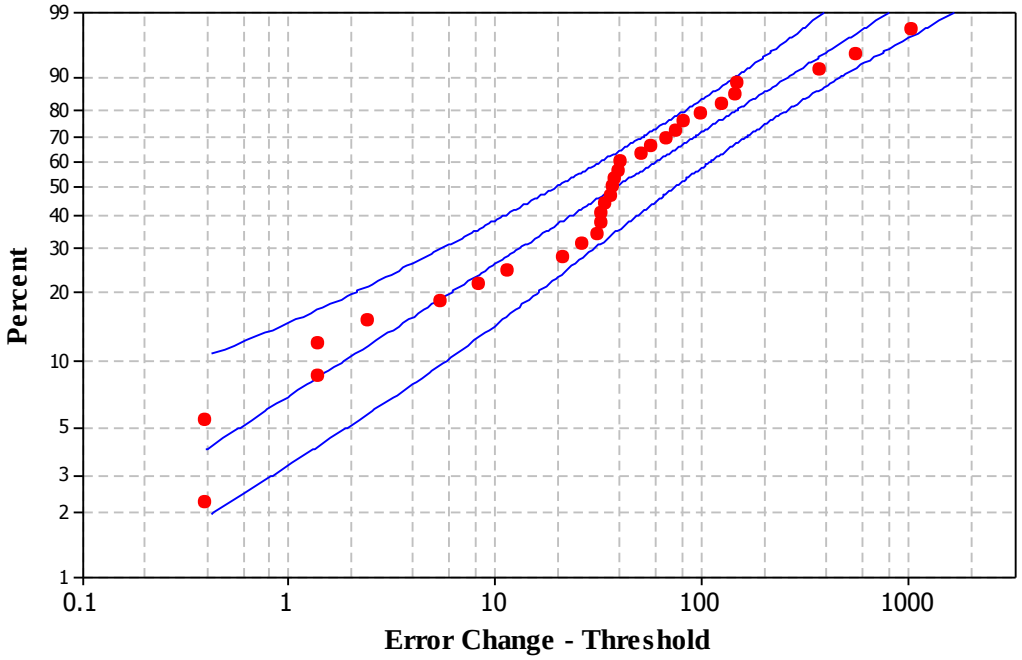

Figure B.38: Plot of Weibull distribution for Error Change in Java Enterprise Edition

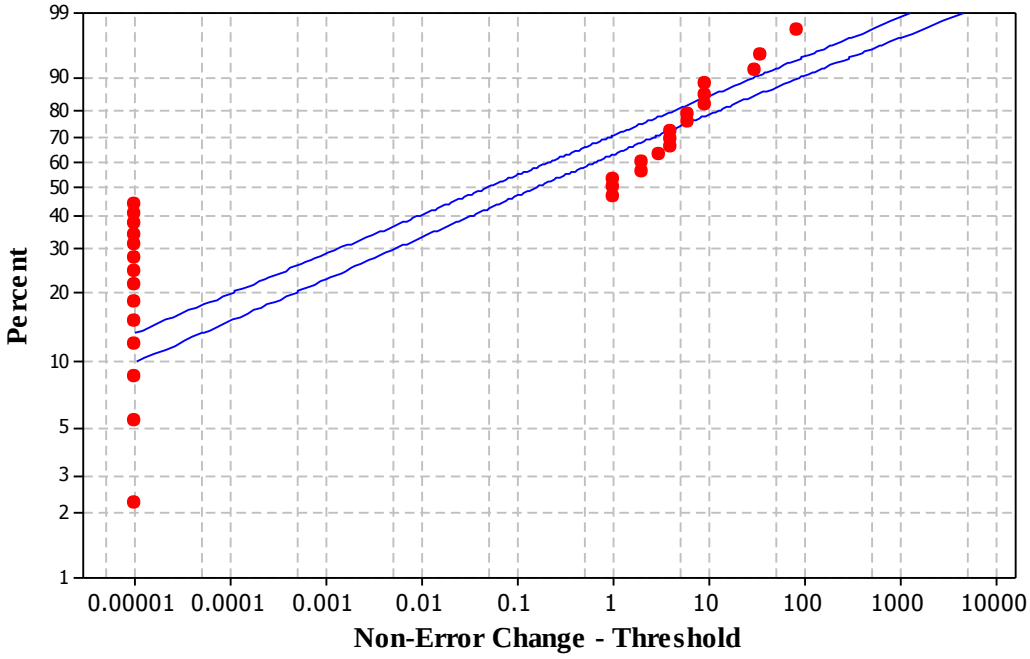

Figure B.39: Plot of Weibull distribution for Non- Error Change in Java Enterprise Edition

Hotspot

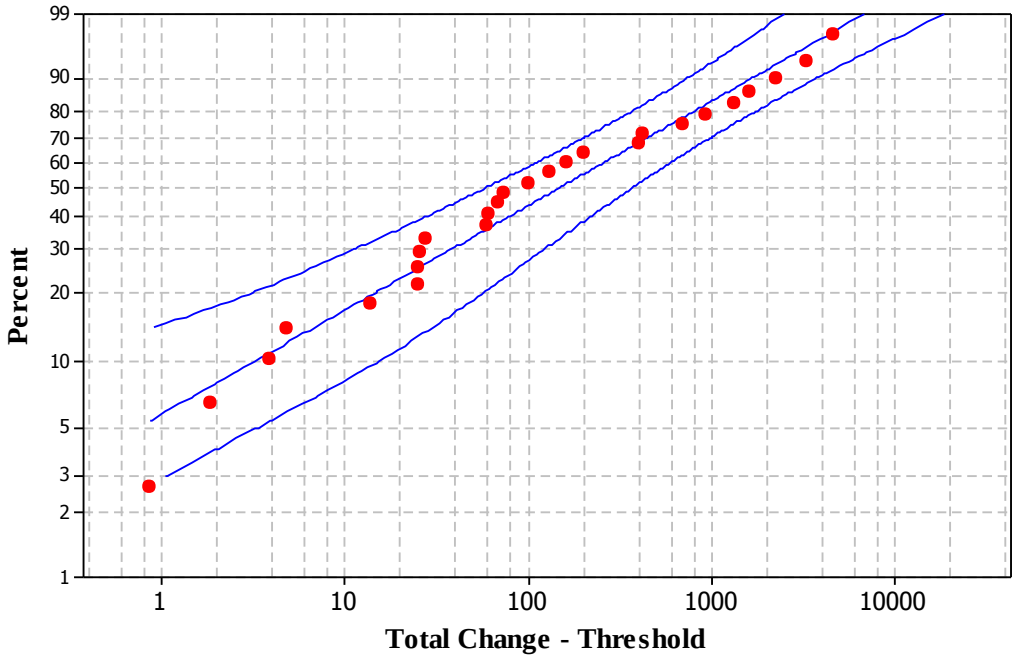

Figure B.40: Plot of Weibull distribution for Total Change in Hotspot

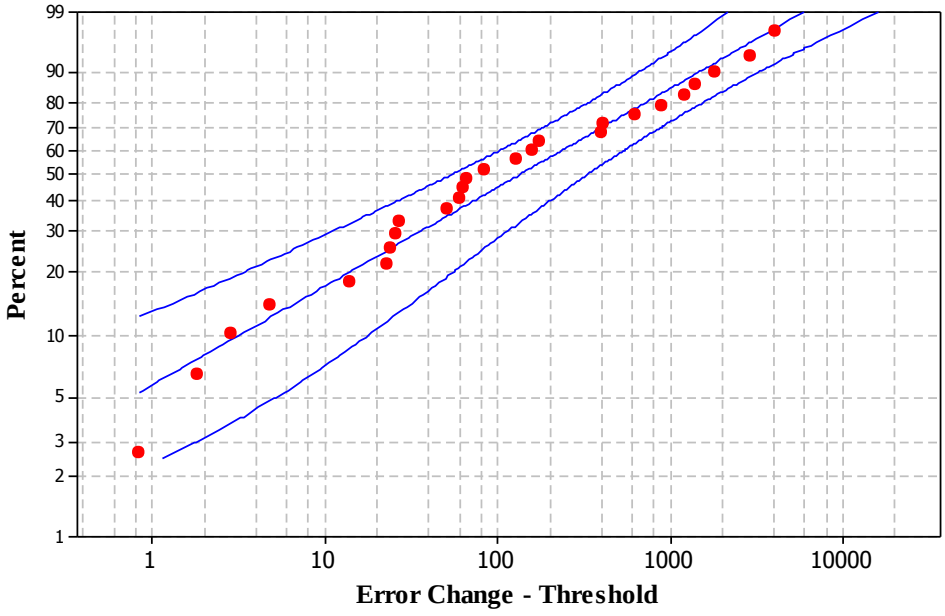

Figure B.41: Plot of Weibull distribution for Error Change in Hotspot

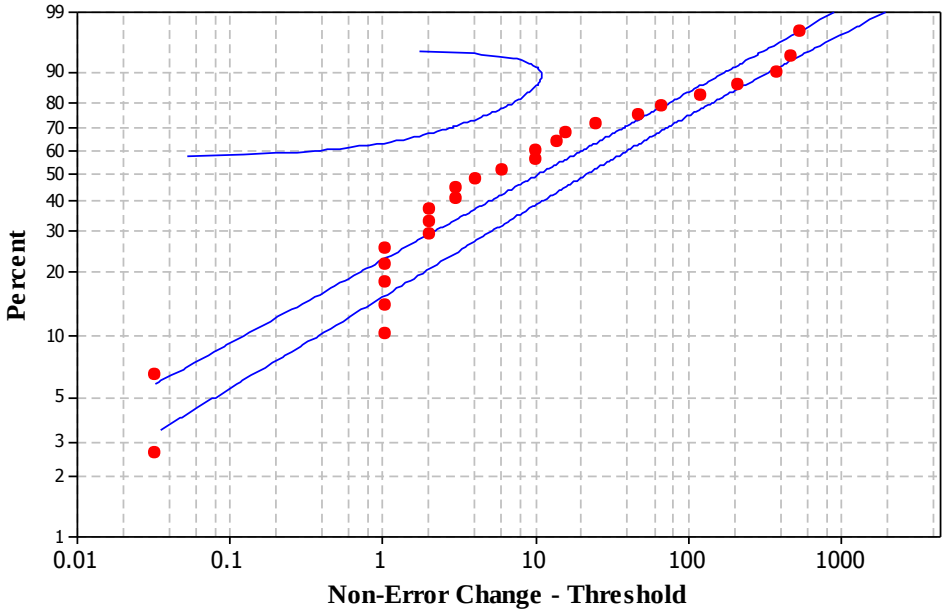

Figure B.42: Plot of Weibull distribution for Non- Error Change in Hotspot

Jiro

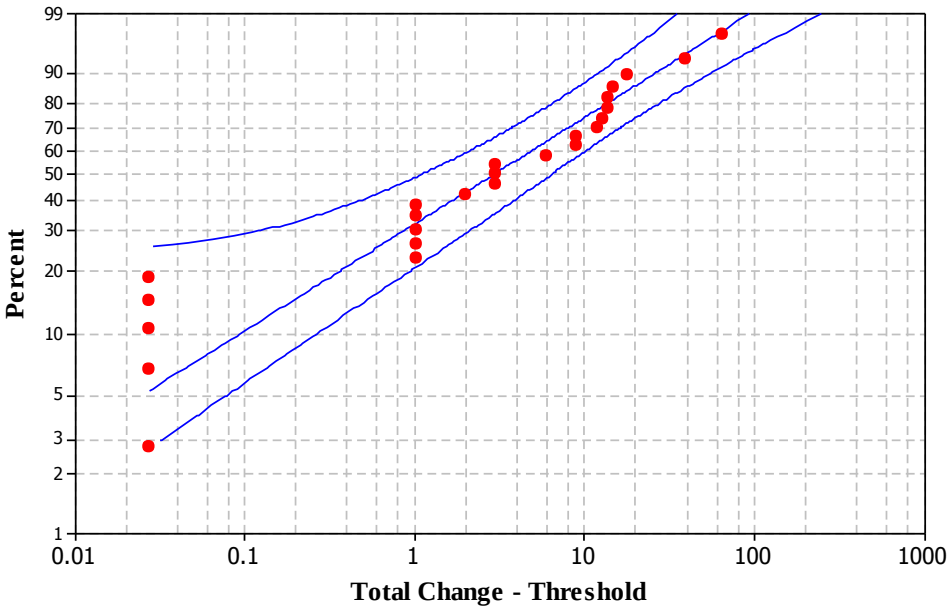

Figure B.43: Plot of Weibull distribution for Total Change in Jiro

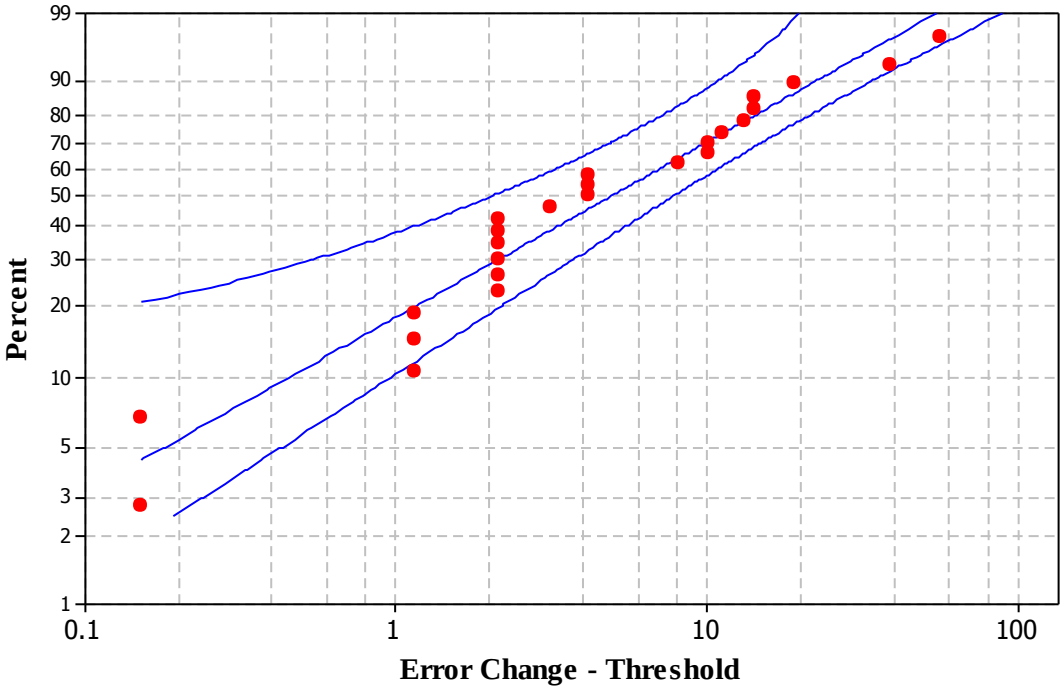

Figure B.44: Plot of Weibull distribution for Error Change in Jiro

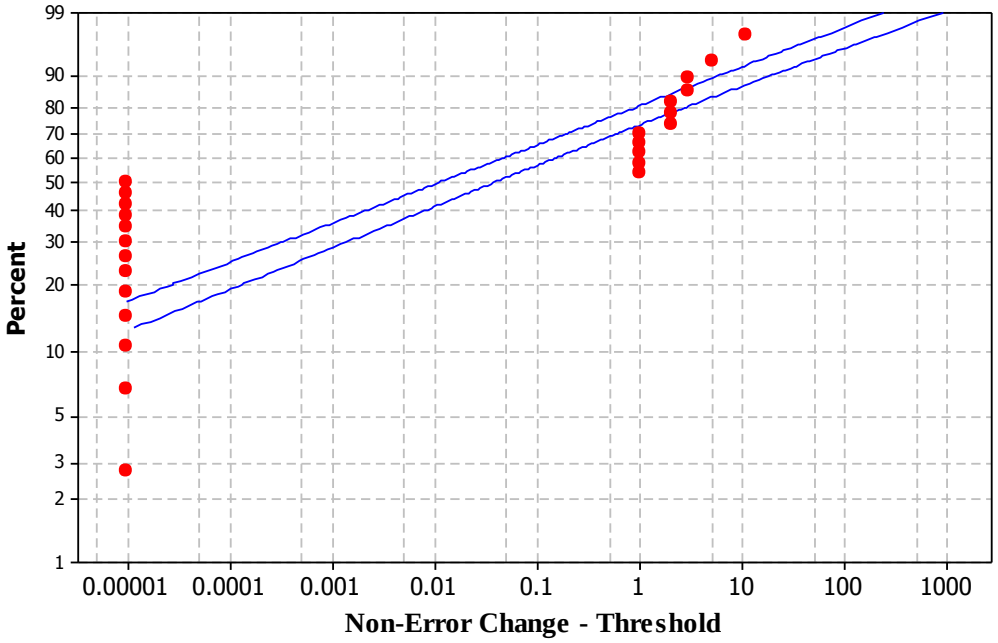

Figure B.45: Plot of Weibull distribution for Non- Error Change in Jiro

Java Web Services Developer Pack

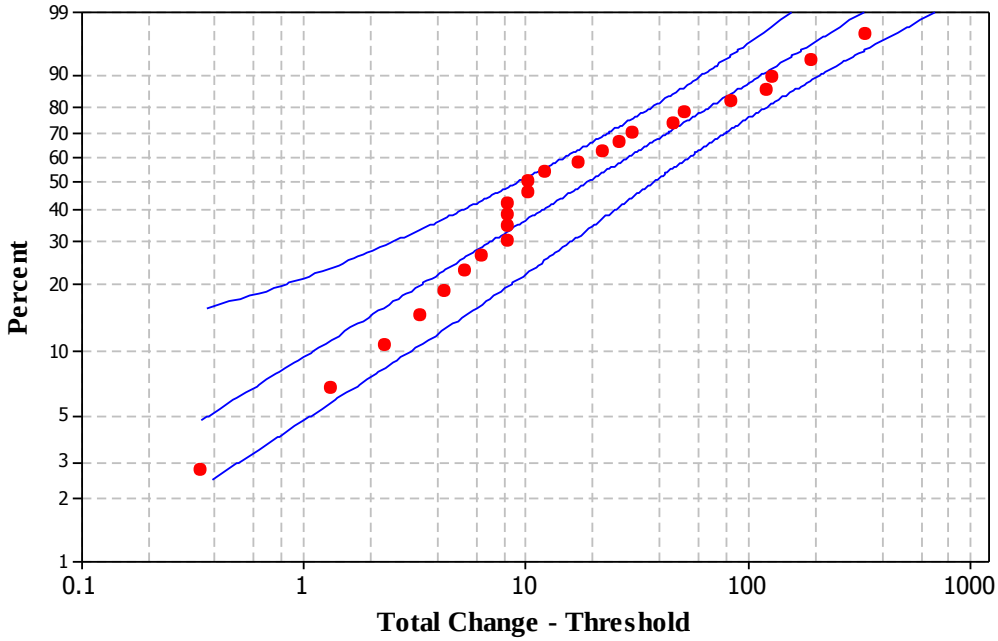

Figure B.46: Plot of Weibull distribution for Total Change in Java Web Services Developer Pack

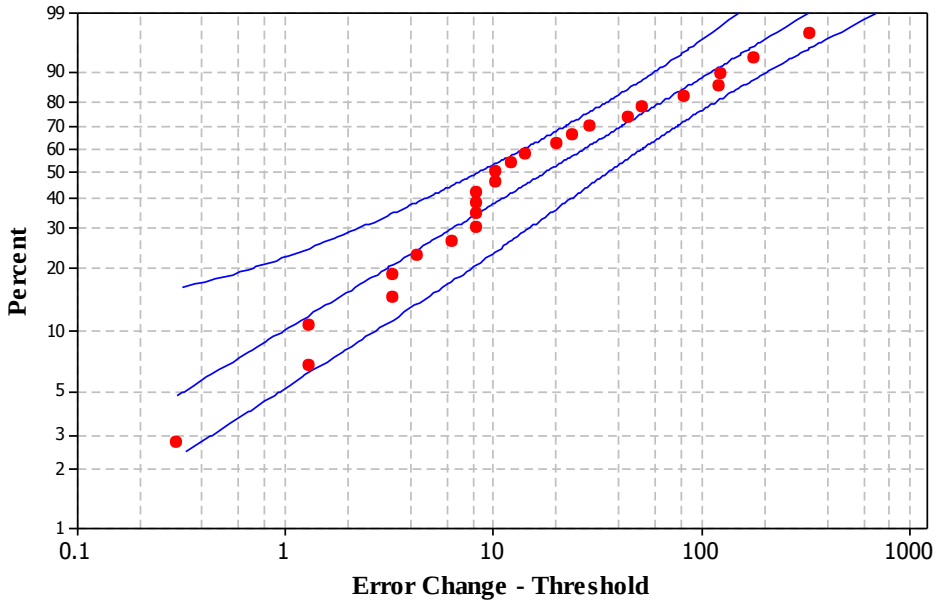

Figure B.47: Plot of Weibull distribution for Error Change in Java Web Services Developer Pack

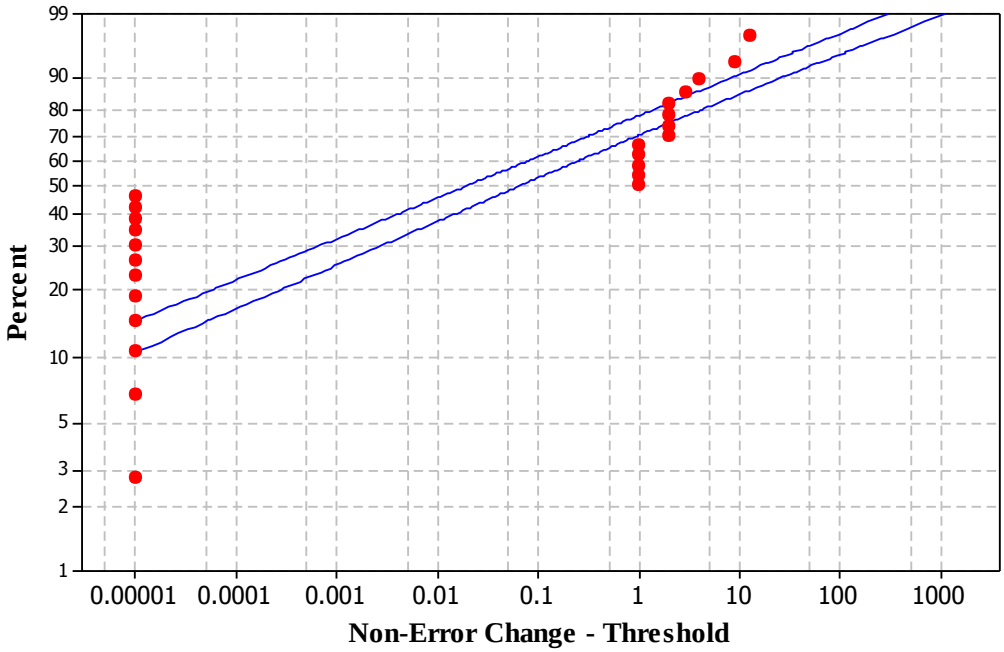

Figure B.48: Plot of Weibull distribution for Non- Error Change in Java Web Services Developer Pack

Java Advanced Imaging

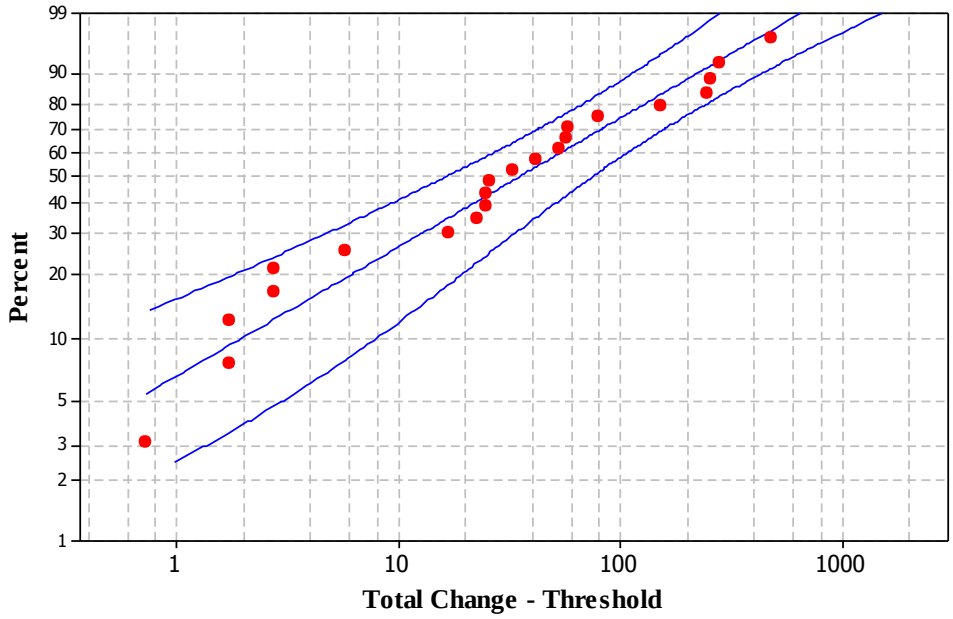

Figure B.49: Plot of Weibull distribution for Total Change Java Advanced Imaging

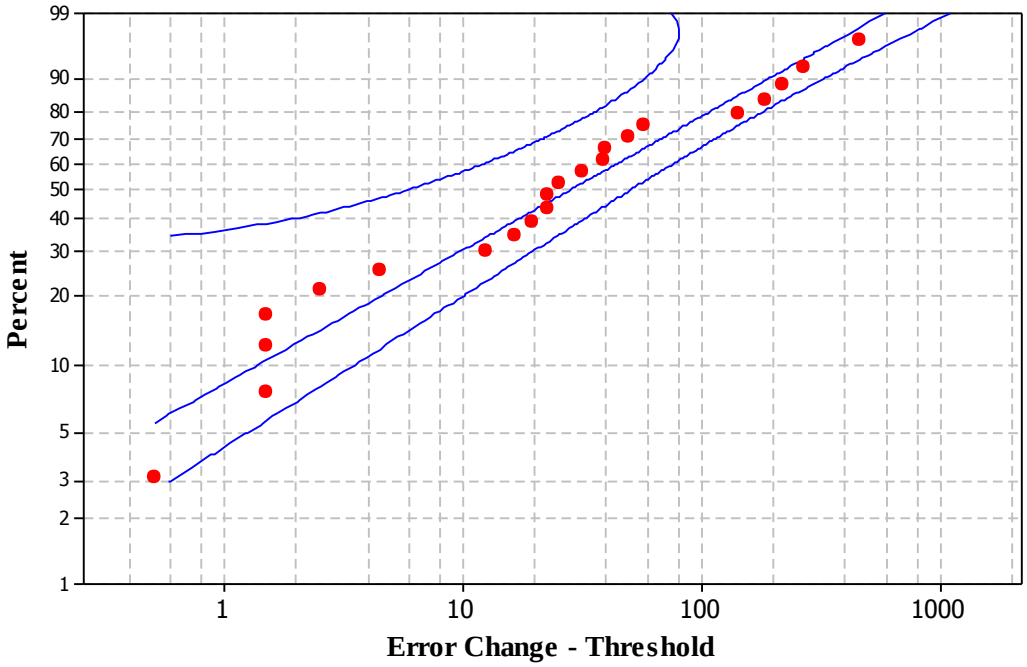

Figure B.50: Plot of Weibull distribution for Error Change Java Advanced Imaging

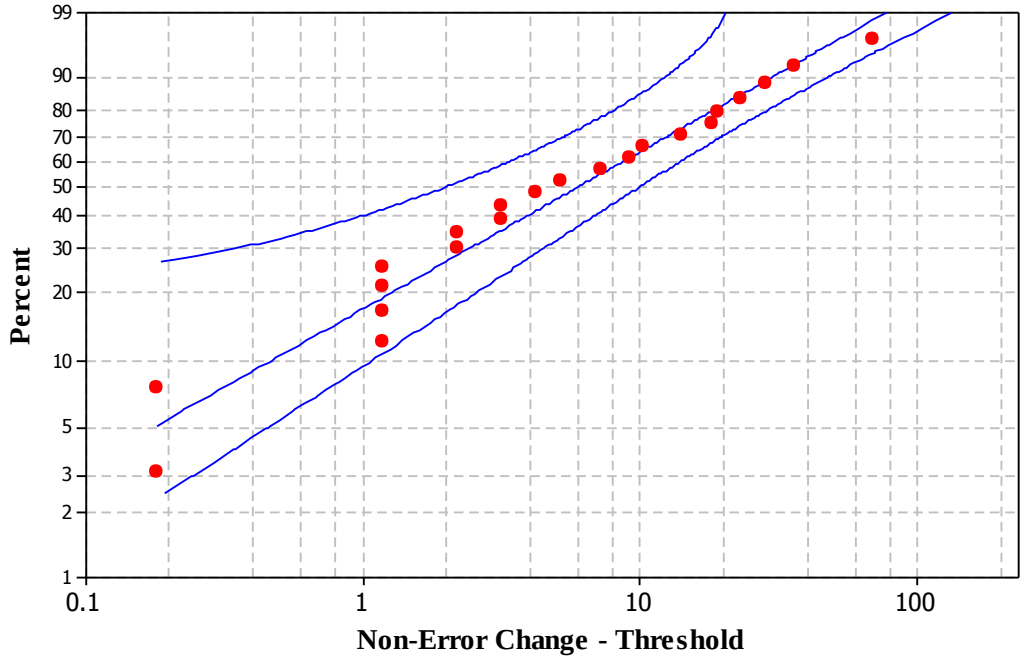

Figure B.51: Plot of Weibull distribution for Non- Error Change Java Advanced Imaging

Portal Server - Search

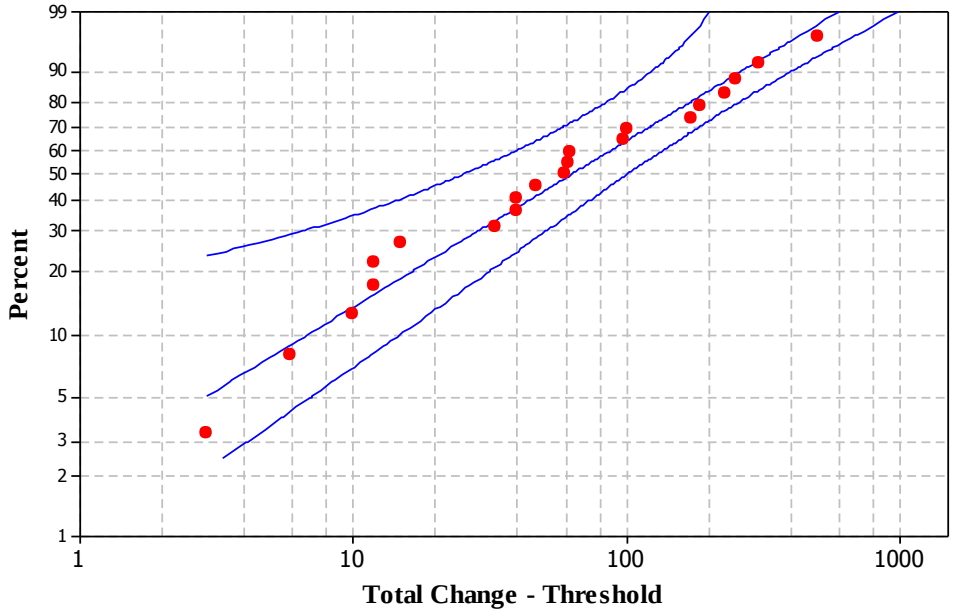

Figure B.52: Plot of Weibull distribution for Total Change in Portal Server - Search

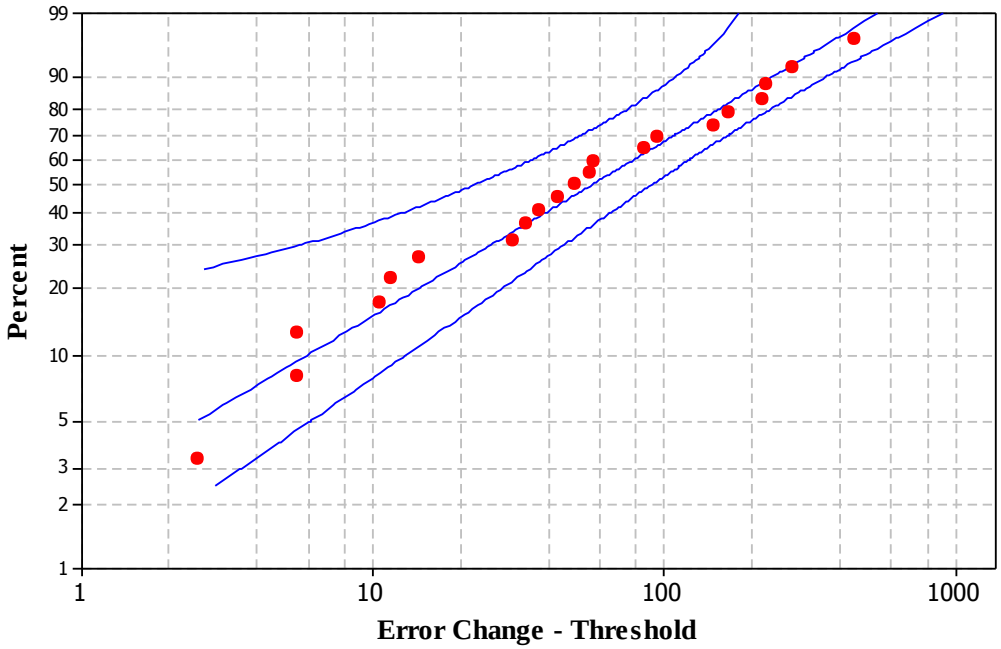

Figure B.53: Plot of Weibull distribution for Error Change in Portal Server - Search

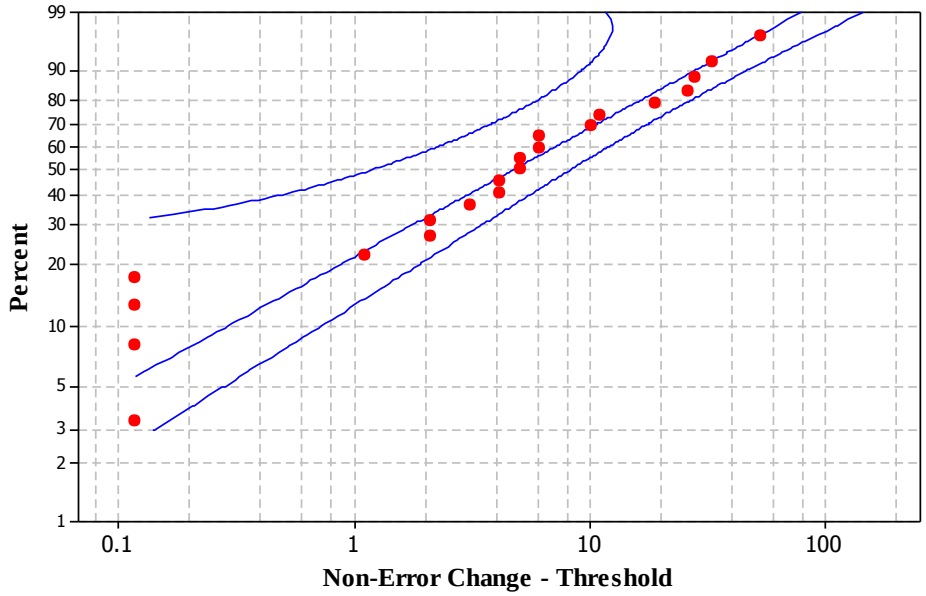

Figure B.54: Plot of Weibull distribution for Non-Error Change in Portal Server - Search

Personal and Embedded Java

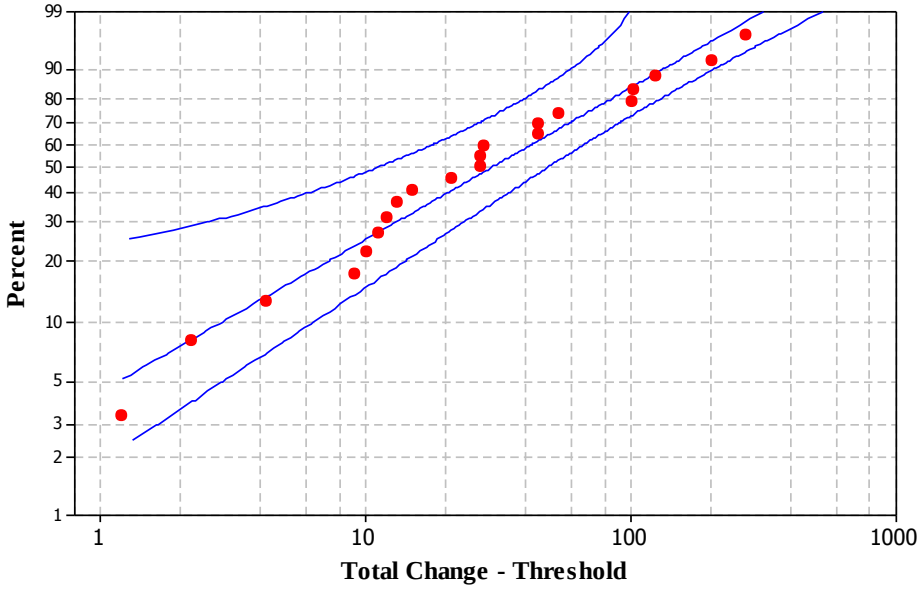

Figure B.55: Plot of Weibull distribution for Total Change in Personal and Embedded Java

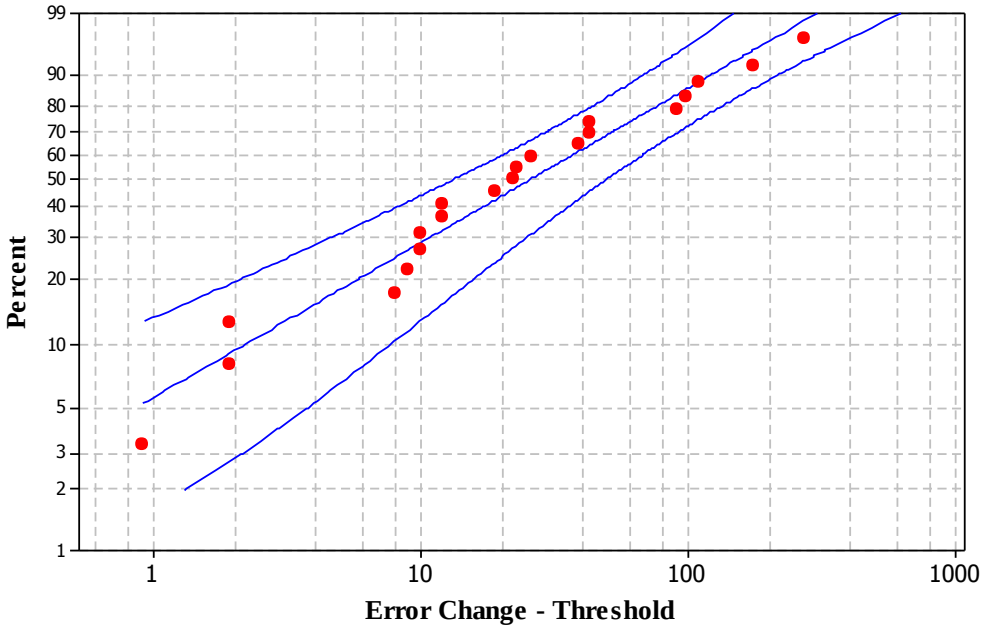

Figure B.56: Plot of Weibull distribution for Error Change in Personal and Embedded Java

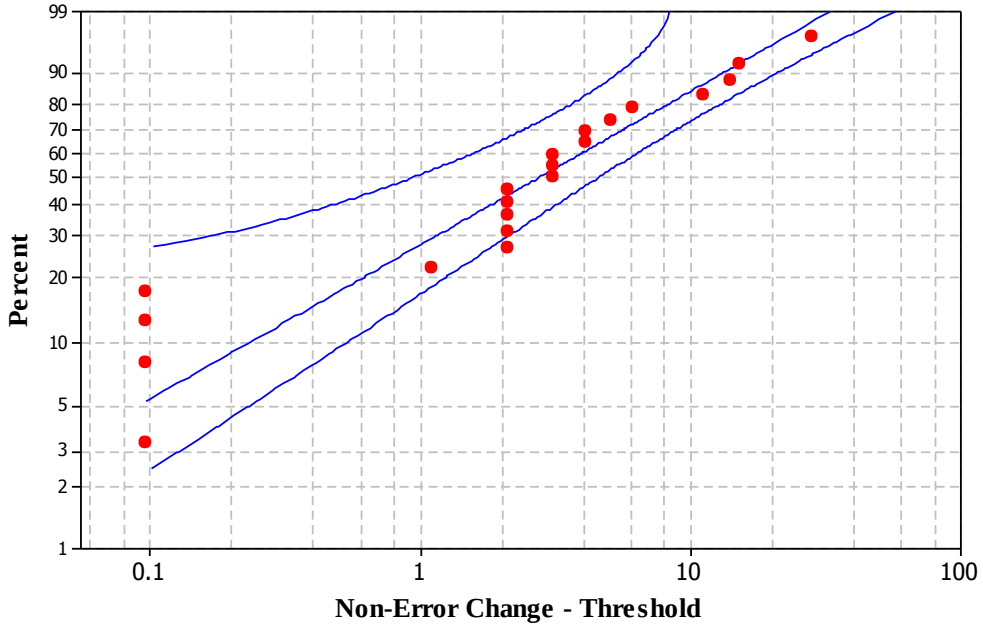

Figure B.57: Plot of Weibull distribution for Error Change in Personal and Embedded Java

Java Plugin

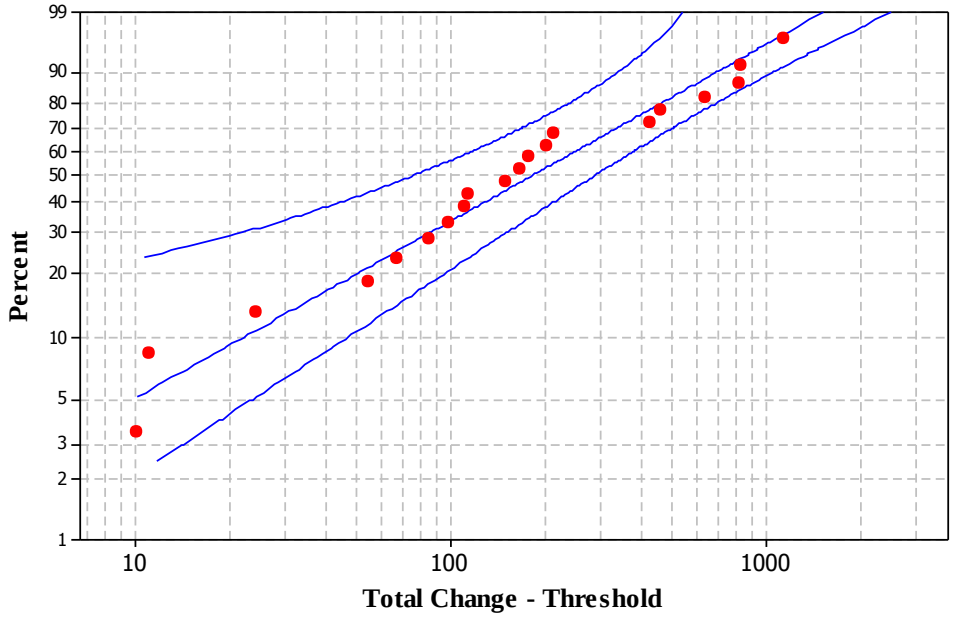

Figure B.58: Plot of Weibull distribution for Total Change in Java Plugin

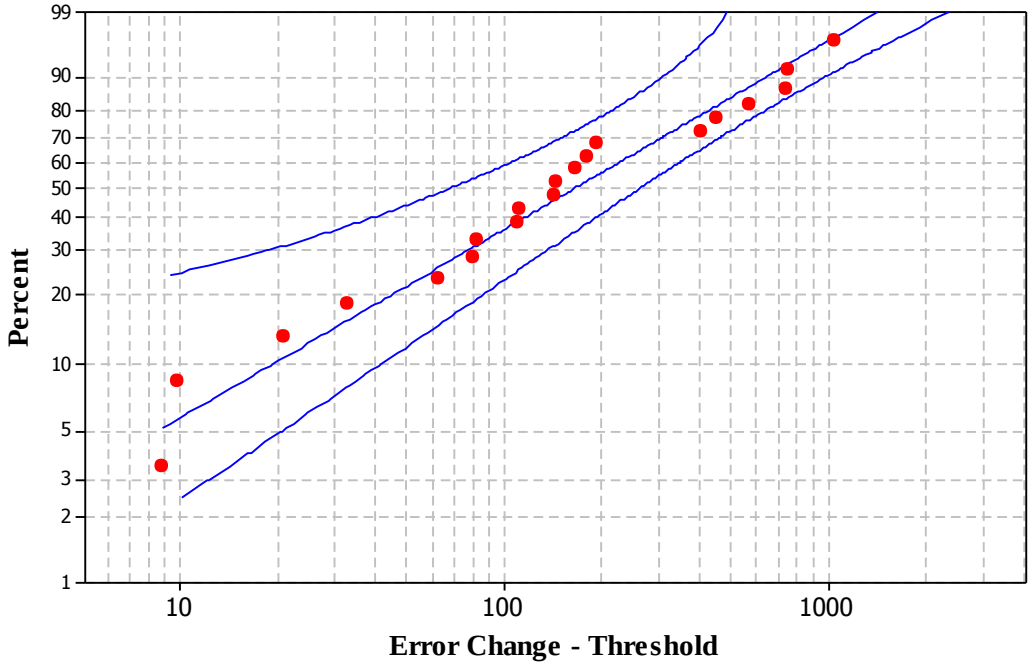

Figure B.59: Plot of Weibull distribution for Error Change in Java Plugin

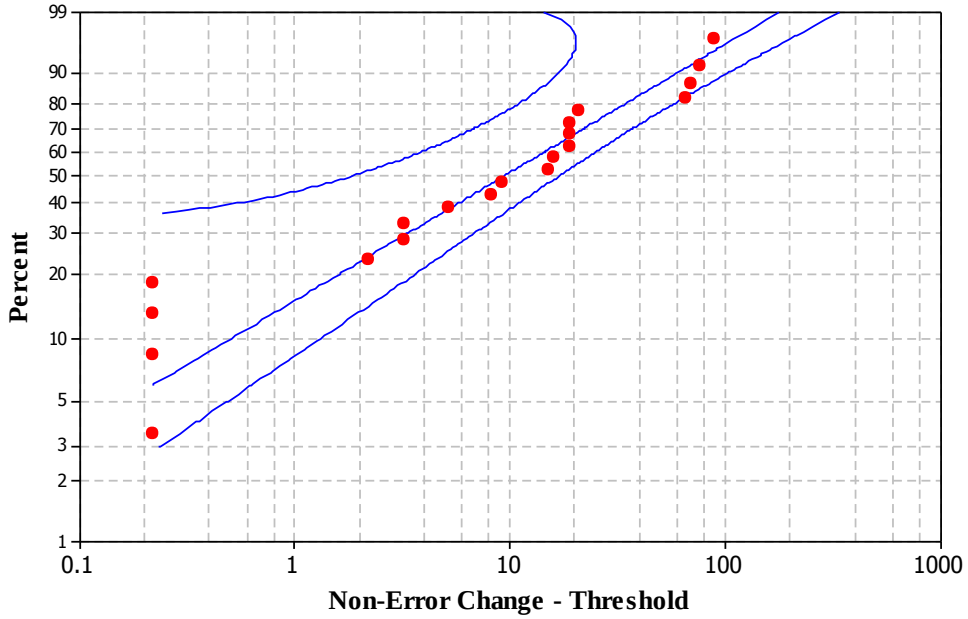

Figure B.60: Plot of Weibull distribution for Non-Error Change in Java Plugin

Java Message Queue

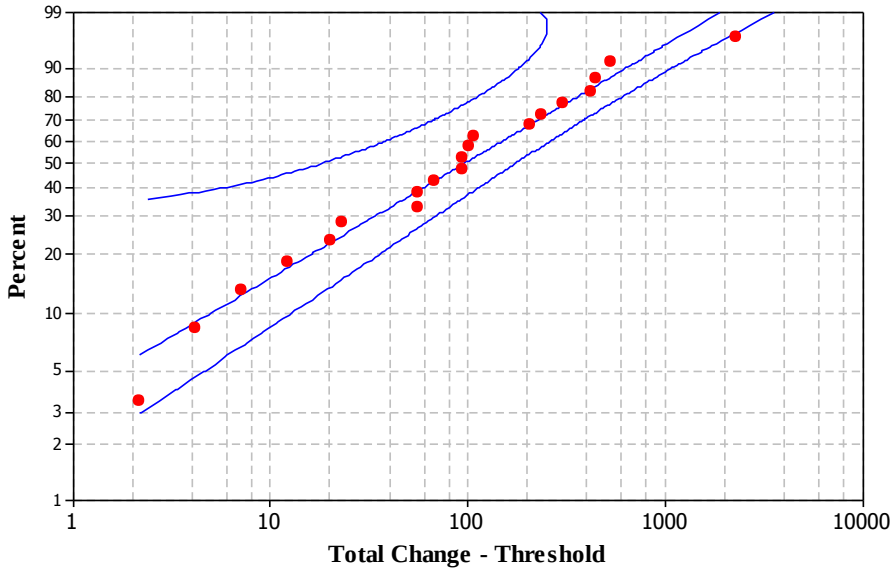

Figure B.61: Plot of Weibull distribution for Total Change in Java Message Queue

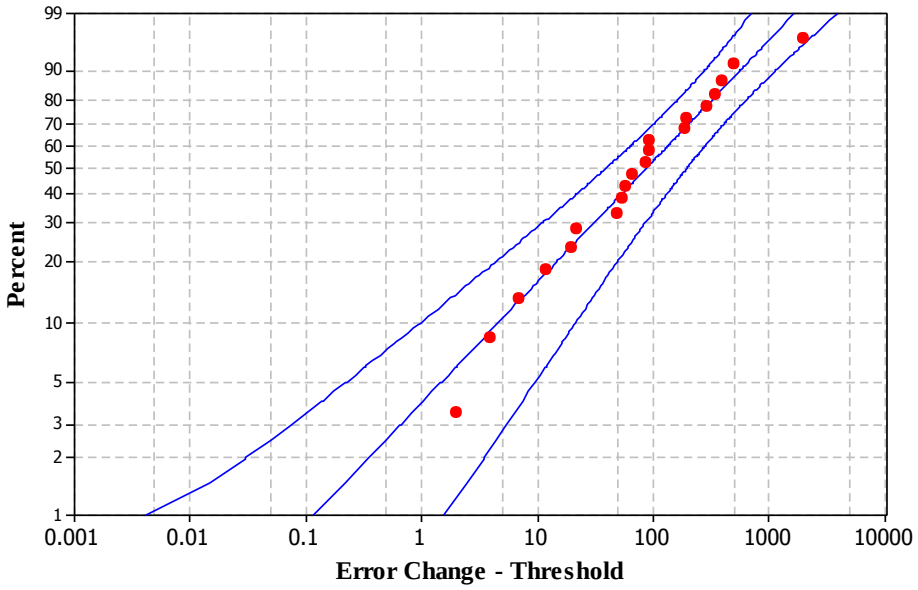

Figure B.62: Plot of Weibull distribution for Error Change in Java Message Queue

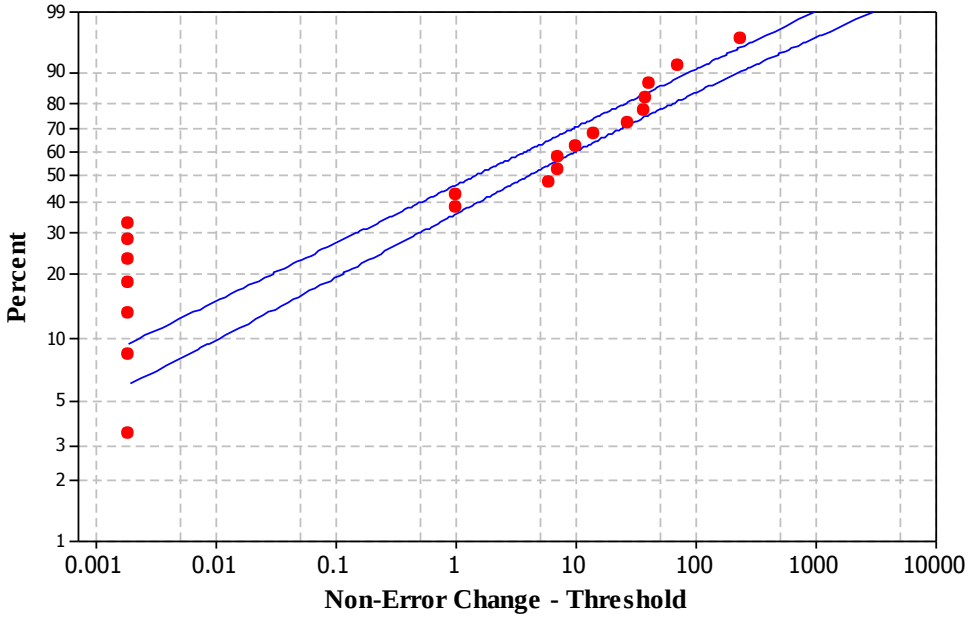

Figure B.63: Plot of Weibull distribution for Non-Error Change in Java Message Queue

Mobile Information Device Profile and Optional Package JSRS - Phoneme Feature

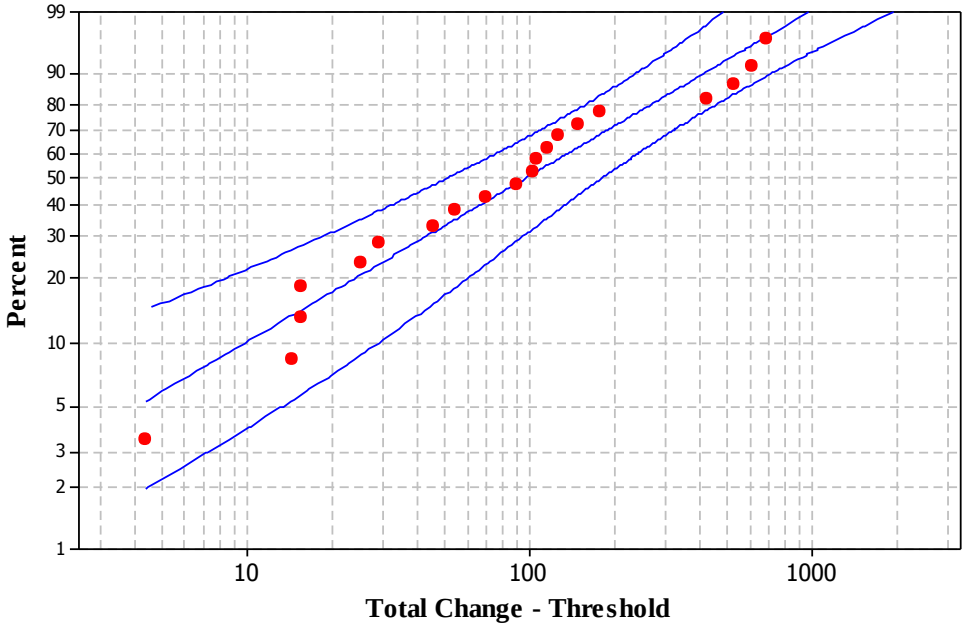

Figure B.64: Plot of Weibull distribution for Total Change in Mobile Information Device Profile and Optional Package JSRS - Phoneme Feature

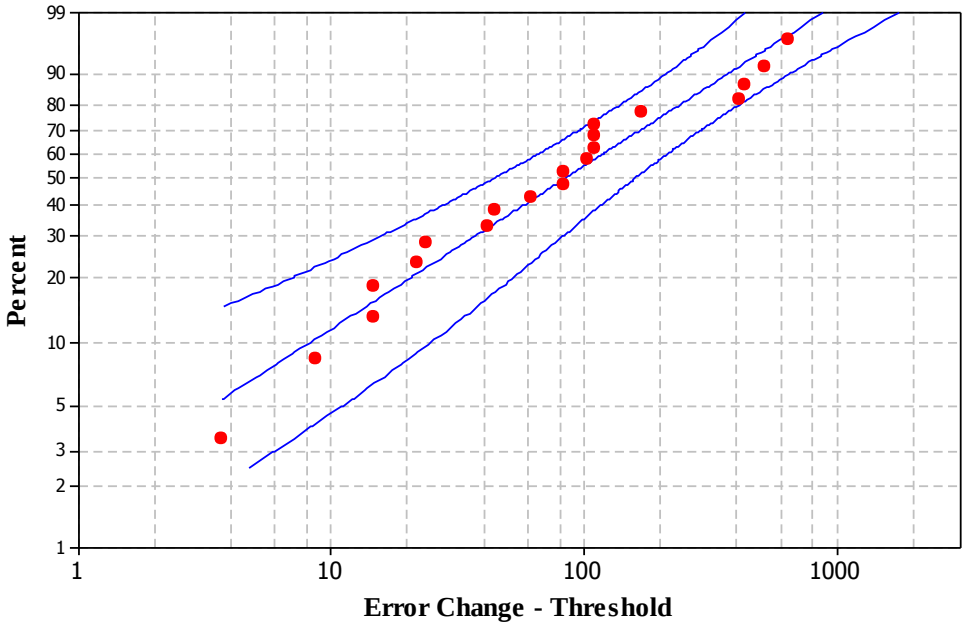

Figure B.65: Plot of Weibull distribution for Error Change in Mobile Information Device Profile and Optional Package JSRS - Phoneme Feature

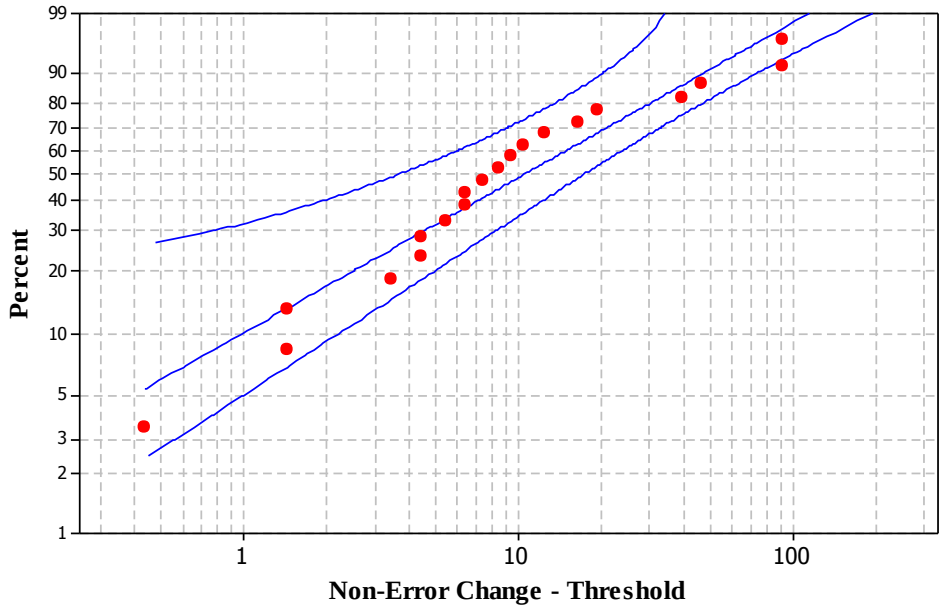

Figure B.66: Plot of Weibull distribution for Non-Error Change in Mobile Information Device Profile and Optional Package JSRS - Phoneme Feature

Java Api For Xml-Based Rpc

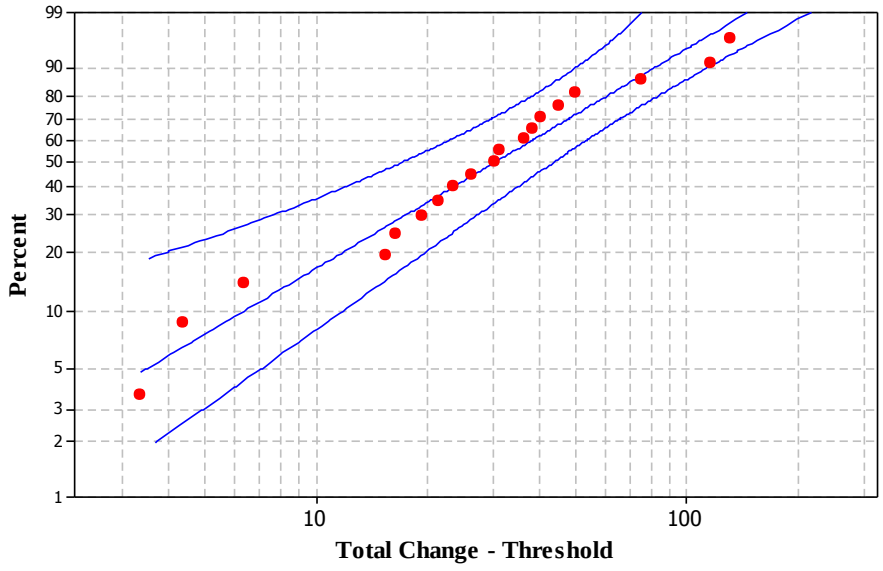

Figure B.67: Plot of Weibull distribution for Total Change in Java Api For Xml-Based Rpc

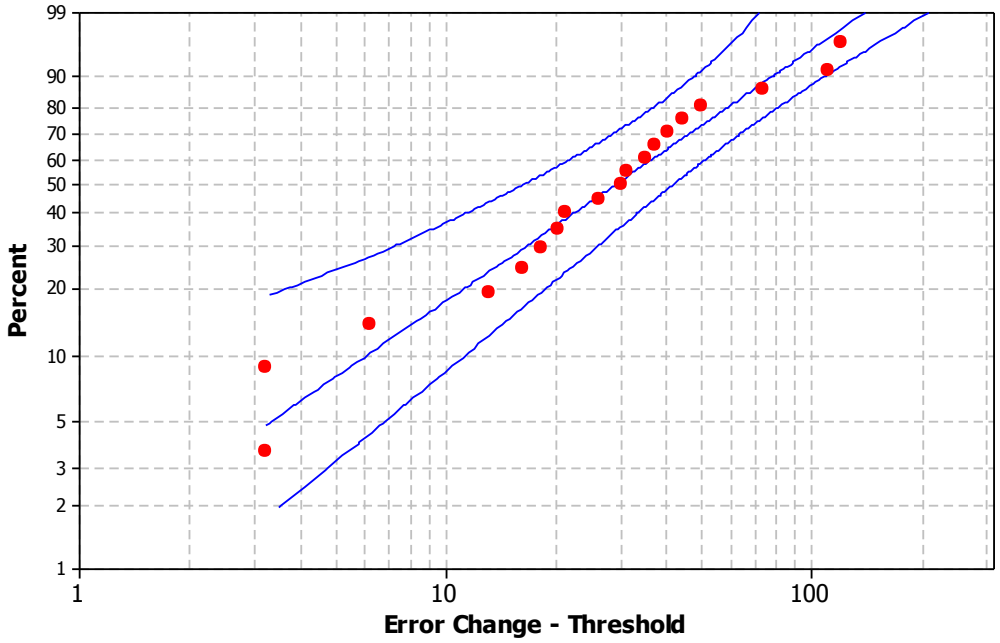

Figure B.68: Plot of Weibull distribution for Error Change in Java Api For Xml-Based Rpc

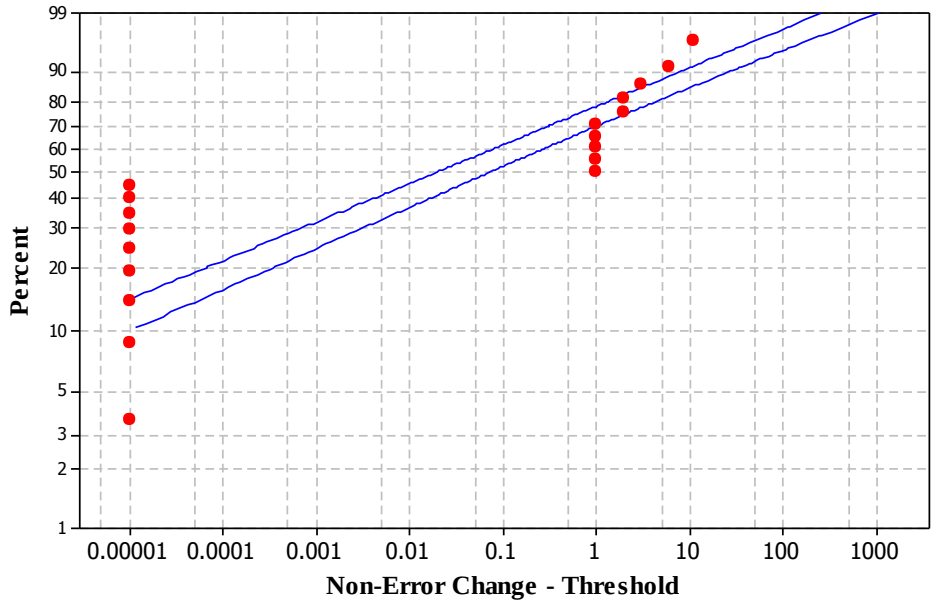

Figure B.69: Plot of Weibull distribution for Non-Error Change in Java Api For Xml-Based Rpc

Java Web Server

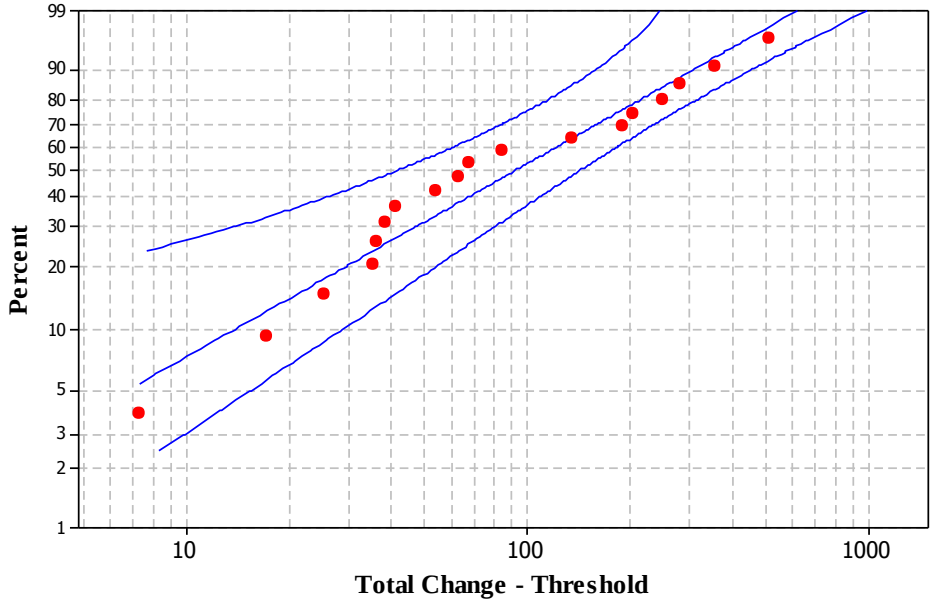

Figure B.70: Plot of Weibull distribution for Total Change in Java Web Server

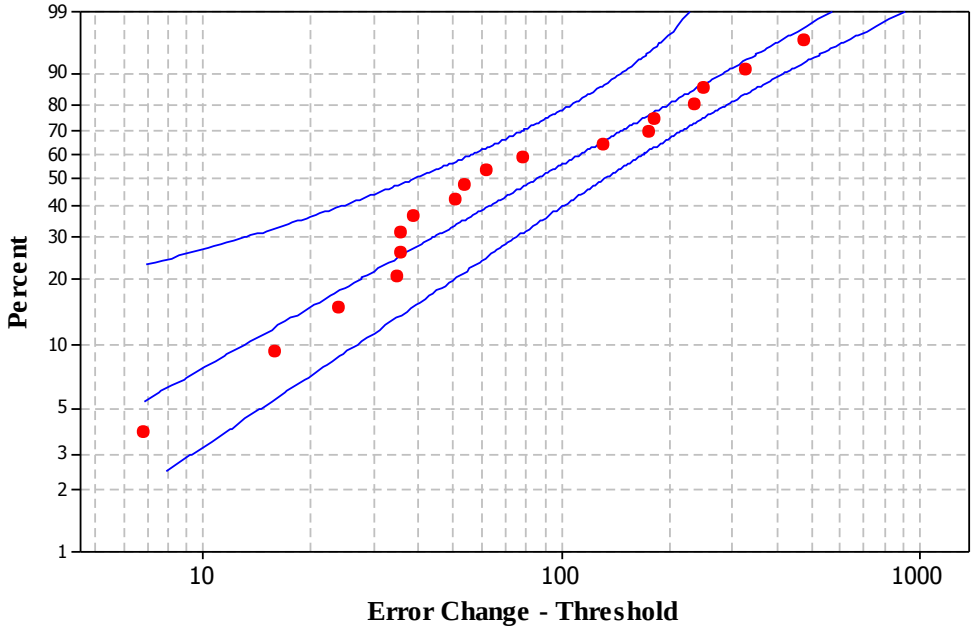

Figure B.71: Plot of Weibull distribution for Error Change in Java Web Server

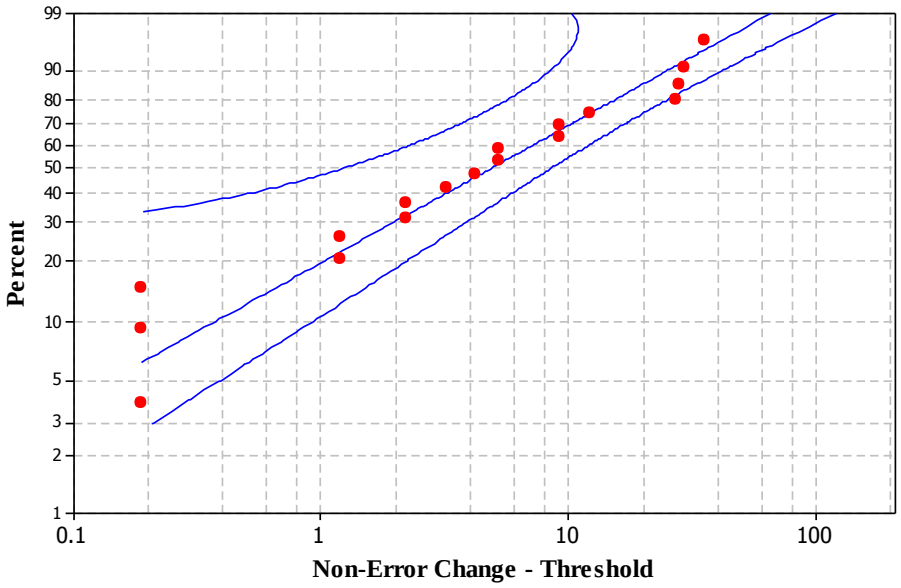

Figure B.72: Plot of Weibull distribution for Non-Error Change in Java Web Server

Mobile Information Device Profile (MIDP) Reference Implementation

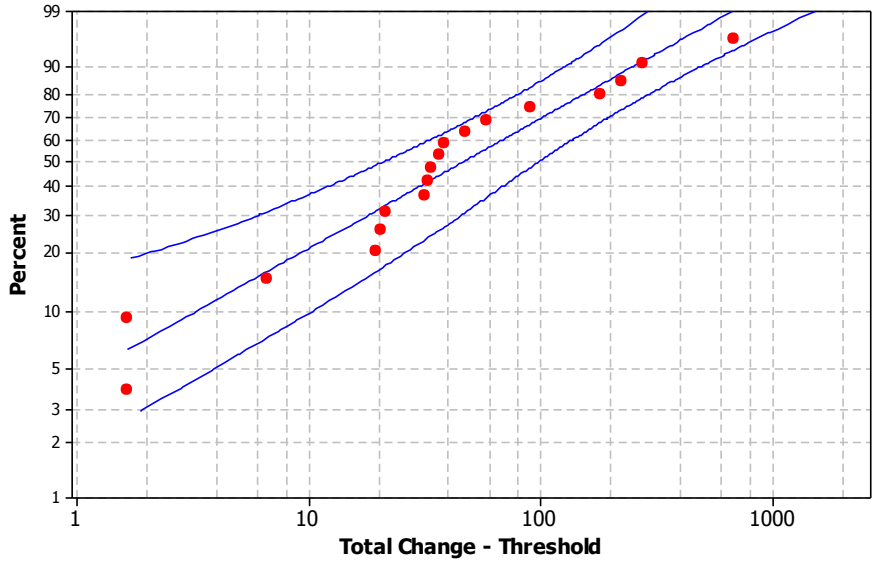

Figure B.73: Plot of Weibull distribution for Total Change in Mobile Information Device Profile (MIDP) Reference Implementation

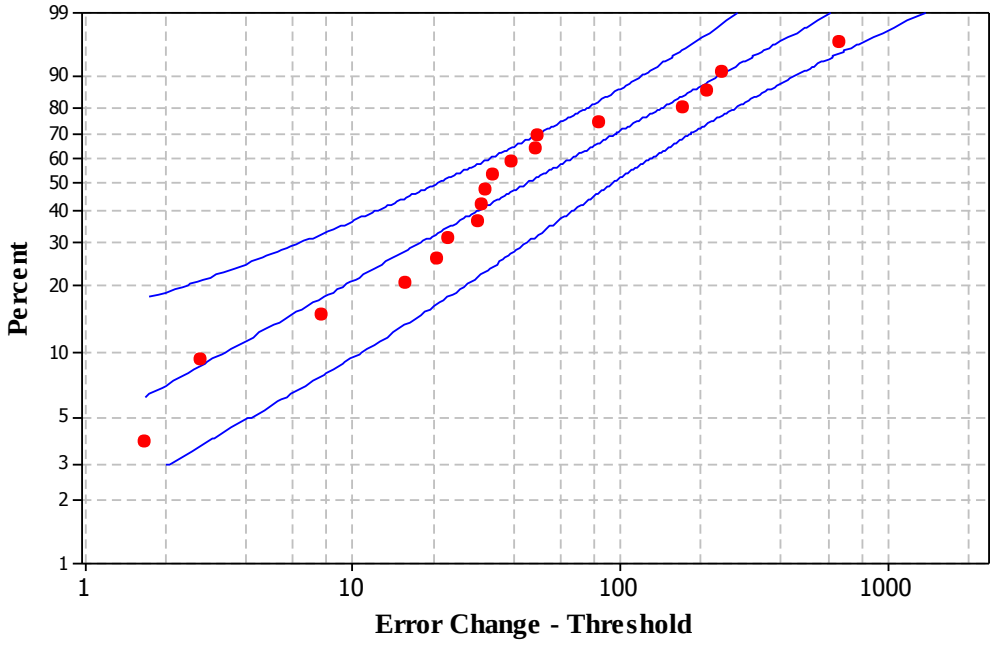

Figure B.74: Plot of Weibull distribution for Error Change in Mobile Information Device Profile (MIDP) Reference Implementation

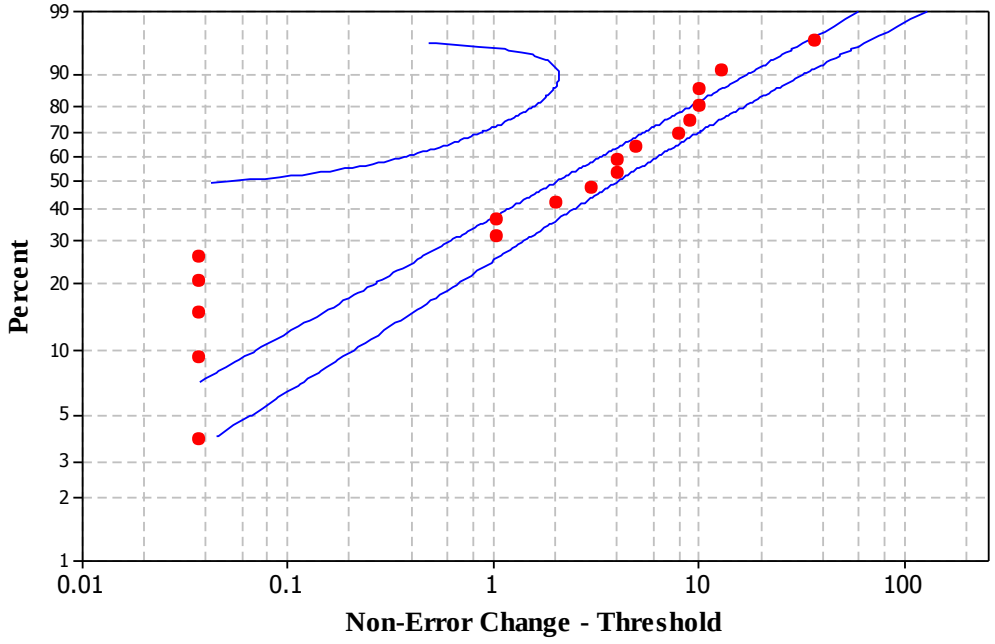

Figure B.75: Plot of Weibull distribution for Non-Error Change in Mobile Information Device Profile (MIDP) Reference Implementation

APPENDIX C

DATA COLLECTION OF FILES CHANGED

C.1 Data Collection of Files Changed

The change data in the NetBeans system were organized in *changesets*. Each changeset consists of atomic changes made to source code repository data. The extraction of changesets involved four steps.

The first step was to download the changelog from the NetBeans Mercurial repository [Mercurial 2011]. This required the installation of the Mercurial software (http://mercurial.selenic.com/wiki/Download) and the creation of the local copy of the NetBeans main repository. A local copy of the NetBeans main repository in the desired path was created from the command line by entering the following command:

hg clone http://hg.netbeans.org/main/ .

After the creation of the local copy of the NetBeans main repository, Mercurial *hg status* command was used to obtain the changed files (changelog) from the downloaded repository. The command to download the changelog and store the information in text file using command line interface is shown below:

hg status >changelog.txt ,

where *changelog.txt* is the filename and the operator '>' redirects the output of command line to the given filename.

The second step was the extraction of the changeset data from the downloaded changelog. A *NetBeans* C*hangeset Parser (NCP)* tool was developed in the C# programming language to collect and process the changeset data from the downloaded changelog. The NCP tool parsed the Mercurial changelog to extract the changeset ID and the summary and store the information in a MYSQL database [MYSQL 2010] on a local machine. Furthermore, for each changeset ID, the NCP tool downloaded the corresponding diff file from "http://hg.netbeans.org/main/rev/" into a text file on the local machine. The changeset ID was appended to the end of the URL to form the complete URL that corresponded to the changeset. A diff file showed the differences between two versions of the same file by showing the number of lines added and removed. Although a changeset might have affected many files such as Java source files, XML files, and HTML files, we only considered the Java source files. The tool grabbed the names of the files modified along with the count of lines added and removed from each changed region of the file. The tool's output was ultimately stored in a database located on the local machine.

The third step was the creation of the input data in the text file format that was required by the H-mine algorithm [Pei et al. 2007]. Philippe [Viger 2011] provided the Java implementation of the H-Mine algorithm that we used for our research. Each row in the input file consists of a collection of unique file IDs (an alias for affected file names in each changeset) separated by white spaces. We illustrate the input and output of the H-mine algorithm with an example in Figure C.1 and Figure C.2, respectively. The numeric digits separated by white spaces in each row represent the collection of file IDs affected by a particular changeset. For the H-mine algorithm in our study, the transaction

database is the collection of affected files associated with each changeset. The changeset ID is the transaction ID and the file IDs are the elements of the transactions. Figure C.1 below shows the input data format for the H-mine algorithm.

```
19280  19283
3336    4766    4768    4771     5013
11160  11181  14880  16994  16995
13726  13752   19280     19283
4110    4118    4122
23876  23877
```

Figure C.1: The Input File Format for the H-Mine Algorithm

To be more specific, sets of files in each row in Figure C.1 are affected by the changeset (the hex string format before colon, as shown in Figure C.2). A collection of changed file IDs are represented in numeric form after ':', separated by white spaces. Figure C.2 below shows the changeset IDs with the changed files IDs.

```
00015dfbe62d: 19280  19283
00016d58ed95: 3336  4766    4768    4771    5013
0004be75ffed: 11160  11181  14880  16994  16995
0005aad717b3: 13726 13752   19280 19283
0006831578e7: 4110   4118   4122
00068376dcc5: 23876    23877
```

Figure C.2: Changeset ID and File ID

The final step was the parsing of the output generated by the H-mine algorithm. The output format of the H-mine algorithm is shown in Figure C.3.

19283:2

19280:2

19280 19283: 2

Figure C.3: Output File Format for H-Mine Algorithm

An H-mine algorithm outputs the complete set of frequent sub patterns for the given input transaction (a collection of logical unit of database operations on a set of items). As shown in Figure C.3, the value before the colon in each row was the set of frequent sub-patterns (set of changed File IDs) and the value after the colon is the frequency of the sub-pattern in the given input transaction (as shown in Figure C.1). The sub-patterns represented the set of elements. The output in Figure C.3 implies that the common subsets of files are affected by the changeset in Figure C.2. The sub-patterns and the frequency information were stored into the database.

APPENDIX D

PARETO PRINCIPLE TEST DATA

Table D.1: NetBeans Source Module and Change Frequency Count

Module Name	Total Frequency
cnd.makeproject	10246
cnd.modelimpl	9521
form	6681
apisupport.project	6562
java.source	5995
subversion	5242
php.editor	5127
cnd.completion	4886
cnd.remote	4490
cnd.debugger.gdb	4329
db	3534
performance	3012
bugzilla	2885
cnd	2867
o.n.core	2807
mercurial	2780
debugger.jpda	2689
cnd.apt	2500
bugtracking	2455
dlight.nativeexecution	2376
java.editor	2355
java.hints	2317
maven	2287
php.project	2284
core.windows	2179
parsing.api	1931
autoupdate.services	1901
cnd.discovery	1877
lib.profiler	1823
kenai.ui	1823
versioning.system.cvss	1814
editor	1800
openide.loaders	1695
vmd.midp	1672

Table D.1 (cont'd…)

nbi	1635
nbbuild	1604
css.editor	1591
debugger.jpda.ui	1534
groovy.editor	1526
web.project	1475
j2ee.persistence	1471
profiler	1466
java.j2seproject	1461
cnd.editor	1457
html.editor	1439
masterfs	1413
refactoring.java	1408
glassfish.common	1344
projectui	1293
installer	1268
autoupdate.ui	1254
openide.explorer	1193
websvc.rest	1179
core.startup	1177
lib.profiler.ui	1142
cnd.debugger.common2	1119
j2ee.ejbcore	1106
core.output2	1061
cnd.api.model	1054
websvc.core	1050
editor.lib2	1047
j2eeserver	1039
editor.lib	1014
web.jsf	1014
web.core.syntax	1004
cnd.gizmo	988
jira	987
properties	947
api.visual	914
cnd.antlr	906
openide.util	899
html.editor.lib	897
junit	892
csl.api	880
openide.filesystems	848
ide.ergonomics	842

Table D.1 (cont'd…)

utilities	841
dlight.remote	839
j2ee.weblogic9	838
cnd.highlight	837
beans	836
jellytools	833
db.dataview	809
openide.nodes	799
spi.debugger.ui	788
glassfish.javaee	782
j2ee.sun.appsrv81	780
j2ee.ejbjarproject	770
vmd.midpnb	766
websvc.wsitconf	761
web.jsf.editor	758
cnd.refactoring	756
o.apache.tools.ant.module	743
project.ant	739
web.jsf.navigation	736
j2ee.ddloaders	719
javascript.editing	712
dlight.visualizers	708
o.n.bootstrap	702
groovy.grailsproject	669
languages	664
spring.beans	663
cnd.repository	663
hibernate	662
java.project	656
git	652
j2ee.sun.ddui	639
cnd.dwarfdiscovery	637
diff	636
cnd.model.services	636
j2ee.common	633
api.debugger	625
ant.freeform	625
dlight	618
kenai	607
bugtracking.bridge	604
web.core	603
hudson	591

Table D.1 (cont'd…)

cnd.debugger.gdb2	587
cnd.dwarfdump	581
dlight.remote.impl	566
cnd.lexer	560
cnd.utils	555
j2ee.earproject	547
websvc.design	547
web.beans	531
cnd.toolchain	514
j2ee.clientproject	514
javadoc	512
mobility.svgcore	511
jemmy	508
i18n	507
tomcat5	501
lexer	501
versioning	500
debugger.jpda.projects	500
lib.profiler.charts	498
dlight.core.stack	497
vmd.game	496
openide.text	494
jmx	484
maven.j2ee	480
o.n.swing.tabcontrol	476
refactoring.api	473
xml.text	465
db.mysql	452
mobility.project	451
cnd.api.remote	448
openide.awt	432
j2ee.jboss4	431
db.metadata.model	427
db.core	425
db.sql.editor	423
java.navigation	422
maven.jaxws	418
uihandler	417
cnd.navigation	414
dlight.perfan	410
dlight.management	406
editor.completion	401
api.debugger.jpda	399

Table D.1 (cont'd…)

php.dbgp	397
j2ee.kit	392
nbjunit	392
maven.model	383
xml.multiview	382
java.j2seplatform	379
ant.debugger	364
clearcase	352
extexecution	348
css.visual	345
projectapi	344
swingapp	343
websvc.saas.api	341
options.editor	327
j2ee.sun.appsrv	326
j2ee.dd	318
websvc.saas.codegen	316
web.client.tools.common	312
ide.kit	311
cnd.classview	311
core.netigso	308
spi.viewmodel	304
localhistory	301
spi.palette	298
api.java.classpath	295
projectuiapi	293
core.ui	292
classfile	292
groovy.grails	289
vmd.components.svg	286
web.jspparser	286
maven.indexer	286
dlight.indicators	283
options.api	281
welcome	281
maven.grammar	280
print	276
java.api.common	276
cnd.modelutil	271
tasklist.ui	269
dlight.annotationsupport	265
editor.codetemplates	265

Table D.1 (cont'd…)

javacard.project	258
dlight.dtrace	256
websvc.kit	250
projectimport.eclipse.core	250
versioning.util	249
xml.schema.completion	249
mobility.end2end	249
o.n.swing.plaf	247
editor.settings.storage	242
web.debug	242
cnd.modelui	230
jumpto	229
xml.schema.model	228
html.parser	222
glassfish.eecommon	215
html	214
api.java	213
websvc.manager	212
apisupport.refactoring	209
form.j2ee	209
vmd.inspector	208
gsf.testrunner	206
websvc.jaxwsmodel	204
core.multiview	201
spi.editor.hints	197
cnd.modeldiscovery	194
settings	193
jellytools.platform	192
options.keymap	190
j2ee.persistenceapi	189
o.n.swing.outline	188
web.client.tools.impl	188
project.libraries	188
web.el	187
extbrowser	182
vmd.model	182
favorites	182
web.monitor	181
java.freeform	179
httpserver	179
debugger.jpda.ant	176
libs.git	174
websvc.wsitmodelext	167

Table D.1 (cont'd…)

spi.quicksearch	166
websvc.saas.ui	165
j2ee.websphere6	164
openide.dialogs	163
maven.embedder	157
websvc.restapi	157
websvc.restkit	156
javahelp	153
derby	149
java.sourceui	148
xml.tax	146
image	145
j2ee.jpa.verification	144
cnd.script	144
cnd.cncppunit	143
openide.windows	140
maven.apisupport	137
maven.graph	136
java.platform	134
xml	134
web.struts	133
cnd.qnavigator	133
websvc.jaxrpc	133
dlight.db.h2	132
html.lexer	131
api.progress	131
cnd.gotodeclaration	130
mobility.jsr172	127
j2ee.metadata.model.support	127
editor.structure	127
xml.xam	125
o.n.swing.dirchooser	124
autoupdate.pluginimporter	123
dlight.threadmap.support	121
schema2beans	121
terminal	121
lib.terminalemulator	118
vmd.properties	117
o.n.upgrader	116
xml.xdm	116
cnd.callgraph	114
db.sql.visualeditor	113

Table D.1 (cont'd…)

xml.jaxb	113
vmd.io	112
php.api.phpmodule	112
editor.indent	111
editor.errorstripe	111
spring.webmvc	110
core.execution	109
xml.catalog	108
php.symfony	107
usersguide	105
editor.mimelookup.impl	105
i18n.form	102
cnd.api.project	101
xml.tools	101
dlight.memory	100
dlight.tools	100
openide.actions	100
groovy.support	100
maven.hints	99
websvc.registry	99
api.xml	98
maven.repository	98
dlight.toolsui	98
openide.util.lookup	98
xml.wsdl.model	97
lib.uihandler	96
dlight.cpu	96
websvc.rest.samples	95
dlight.util	95
javawebstart	95
j2ee.api.ejbmodule	94
profiler.snaptracer	94
cnd.debugger.common	94
debugger.jpda.heapwalk	93
dlight.collector.stdout	92
lib.profiler.common	91
api.web.webmodule	91
xml.core	91
vmd.componentssupport	91
java.kit	89
reglib	88
xsl	88

Table D.1 (cont'd…)

xml.retriever	88
wag.manager	87
html.validation	86
j2ee.sun.dd	85
php.refactoring	84
core.browser	84
xml.axi	84
j2ee.ejbverification	82
editor.fold	81
websvc.clientapi	81
web.refactoring	81
vmd.screen	81
cnd.simpleunit	81
dlight.sync	79
editor.bookmarks	79
javacard.ri.platform	78
jsp.lexer	77
dbschema	77
bugtracking.kenai	76
spi.navigator	76
server	76
j2ee.core.utilities	75
web.common	75
mobility.editor	75
editor.util	74
web.freeform	73
dlight.extras	73
o.openidex.util	72
cnd.source	68
websvc.wsstack.jaxws	67
websvc.rest.wadl.design	67
websvc.saas.codegen.j2ee	65
java.source.ant	65
languages.yaml	65
javascript.refactoring	64
jellytools.ide	63
dlight.db.derby	60
dlight.procfs	59
cnd.testrunner	57
javascript.hints	57
jellytools.enterprise	55
apisupport.crudsample	55

Table D.1 (cont'd…)

openide.io	54
cnd.folding	54
ant.grammar	54
vmd.io.javame	54
spellchecker	53
editor.settings	53
groovy.gsp	53
cnd.asm	52
profiler.j2ee	51
websvc.jaxwsapi	51
websvc.customization	50
j2ee.archive	50
dlight.terminal	50
j2ee.jpa.refactoring	50
progress.ui	50
collab.ui	50
core.io.ui	50
editor.bracesmatching	49
j2ee.samples	49
identity.profile.ui	49
queries	47
vmd.palette	46
websvc.saas.codegen.java	46
php.zend	45
editor.mimelookup	45
core.nativeaccess	44
keyring	43
java.debug	43
j2me.cdc.project	43
javacard.spi	42
apisupport.installer	42
collab.channel.filesharing	42
parsing.lucene	41
mobility.cldcplatform	41
java.lexer	41
web.kit	41
hudson.subversion	40
cnd.repository.api	40
websvc.jaxws.lightapi	40
maven.junit	40
tasklist.todo	39
vmd.flow	39

Table D.1 (cont'd…)

dlight.core.ui	38
applemenu	38
jconsole	38
core.kit	38
websvc.projectapi	38
java.editor.lib	37
lib.cvsclient	37
profiler.projectsupport	37
vmd.midp.converter	36
groovy.kit	36
jellytools.java	36
websvc.axis2	35
openide.execution	34
profiler.oql	34
profiler.oql.language	34
maven.profiler	34
j2ee.ant	33
hudson.mercurial	32
apisupport.feedreader	32
javascript.libraries	31
timers	31
spi.tasklist	31
websvc.saas.codegen.php	31
profiler.j2se	31
libs.bugtracking	31
vmd.codegen	31
maven.osgi	30
mobility.midpexamples	30
j2ee.dd.webservice	30
projectimport.jbuilder	29
sendopts	29
dbapi	28
web.client.tools.firefox	28
mobility.j2meunit	27
apisupport.osgidemo	26
websvc.websvcapi	26
libs.freemarker	26
mobility.antext	25
websvc.editor.hints	25
dlight.spi	24
websvc.rest.wadl.model	23
el.lexer	23
maven.persistence	22

Table D.1 (cont'd…)

tasklist.projectint	22
j2me.cdc.platform	21
keyring.impl	21
java.examples	20
projectimport.eclipse.j2se	20
javacard.filemodels	20
dlight.fops	20
core.osgi	19
dlight.collector.procfs	19
xml.tools.java	19
profiler.j2ee.tomcat	19
dlight.msa	19
profiler.selector.ui	19
projectimport.eclipse.web	19
web.examples	19
hudson.maven	19
editor.macros	18
javacard.common	18
collab.provider.im	18
core.browser.xulrunner	18
bugzilla.exceptionreporter	17
editor.guards	16
websvc.metro.samples	16
j2me.cdc.project.ricoh	16
vmd.analyzer	15
jellytools.cnd	15
lexer.nbbridge	15
xml.lexer	15
dlight.libs.common	15
openide.modules	15
api.mobility	15
core.ide	14
properties.based.dataobjects	14
gototest	14
web.client.tools.internetexplorer	14
websvc.utilities	14
maven.samples	14
maven.search	14
web.jsfapi	13
swing.validation	13
utilities.project	13
web.client.tools.api	13
spellchecker.bindings.htmlxml	12

Table D.1 (cont'd…)

languages.ini	12
cnd.spellchecker.bindings	12
profiler.j2ee.sunas	12
wag.codegen	12
o.apache.jmeter.module	12
gsf.codecoverage	12
hibernateweb	12
cnd.kit	11
editor.indent.project	11
kenai.maven	11
web.fake	11
profiler.nbmodule	11
spi.actions	11
apisupport.paintapp	11
profiler.attach	10
groovy.refactoring	10
identity.profile.api	10
profiler.selector.java	10
dlight.webstack	10
vmd.structure	10
o.n.insane	10
identity.kit	10
wag.codegen.java	9
websvc.jaxwsmodelapi	9
editor.deprecated.pre65formatting	9
editor.actions	9
j2ee.metadata	9
ide	9
j2me.cdc.project.savaje	9
java.hints.processor	9
php.phpdoc	9
apisupport.harness	8
j2ee.core	8
versioning.kenai	8
jmx.common	8
j2me.cdc.project.bdj	8
websvc.saas.kit	7
ant.browsetask	7
mobility.deployment.nokia	7
mobility.cldcplatform.catalog	7
editor.plain	7
mobility.deployment.ricoh	7
web.client.javascript.debugger.ant	7

Table D.1 (cont'd…)

j2me.cdc.project.nsicom	7
openide.compat	6
j2ee.genericserver	6
cnd.litemodel	6
profiler.j2ee.generic	6
profiler.selector.spi	6
profiler.j2ee.weblogic	6
php.samples	6
updatecenters	6
javaee.beanvalidation	6
j2me.cdc.project.semc	6
mobility.plugins.mpowerplayer	5
properties.syntax	5
libs.svnClientAdapter	5
o.mozilla.rhino.patched	5
mobility.deployment.sonyericsson	5
identity.samples	5
profiler.freeform	5
web.primefaces	5
openide.options	5
vmd.components.midp	5
collab.channel.output	4
java.preprocessorbridge	4
wag.codegen.j2ee	4
apisupport.ant	4
maven.spring	4
deployment.deviceanywhere	4
c.s.collablet.moxc	3
profiler.j2ee.jboss	3
websvc.saas.services.strikeiron	3
mobility.project.ant	3
websvc.wsstackapi	3
spellchecker.bindings.properties	3
collab.channel.chat	3
target.iterator	3
j2me.cdc.platform.nsicom	3
identity.ant	3
libs.aguiswinglayout	3
libs.nbi.ant	3
deployment.wm	3
javacard.oberthur	3
wag.codegen.php	3
loadgenerator	3

Table D.1 (cont'd…)

hudson.ant	3
javacard.platform.ui	2
uihandler.interactive	2
profiler.utilities	2
j2me.cdc.project.sjmc	2
versioning.indexingbridge	2
j2me.cdc.project.nokiaS80	2
websvc.metro.model	2
javacard.ri.bundle	2
print.editor	2
javacard.console	2
languages.manifest	2
editor.plain.lib	2
languages.refactoring	2
dlight.msa.support	2
spellchecker.bindings.java	2
mobility.deployment.ftpscp	2
websvc.saas.services.facebook	2
groovy.samples	2
dlight.threads	2
c.s.collablet	1
libs.svnClientAdapter.javahl	1
libs.jira	1
o.apache.tools.ant.module.docs	1
maven.coverage	1
profiler.attach.impl	1
websvc.saas.services.flickr	1
identity.server.manager	1
languages.diff	1
j2ee.ejbrefactoring	1
j2me.cdc.platform.bdj	1
j2me.cdc.platform.nokias80	1
mobility.licensing	1
mobility.deployment.webdav	1
j2me.cdc.project.execuiimpl	1

REFERENCES

[Adamic 2011] L.A. Adamic,"Zipf, Power-laws, and Pareto - A ranking tutorial," http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html, 2011.

[Agrawal and Srikant 1994] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," In Proceedings of the 20th Very Large Data Bases Conference (VLDB), pp. 487–499, 1994.

[Alias-i 2008] Alias-i. 2008. LingPipe 4.0.1. http://alias-i.com/lingpipe

[Anderson and Runeson 2007] C. Anderson and P. Runeson, "A replicated quantitative analysis of fault distributions in complex software systems," IEEE Transaction on Software Engineering vol. 33, no. 5, pp. 273-286, 2007.

[Beizer 1990] B. Beizer, Software Testing Techniques, Second Edition, ISBN: 1850328803, 1990.

[Bennett and Rajlich 2000] K. H. Bennett and V. Rajlich, "Software maintenance and evolution: A roadmap," ICSE'2000 – Future of Software Engineering, pp. 73-87, 2000.

[Blei et al. 2003] D.M. Blei, A.Y.Ng, and M.I.Jordan, "Latent Dirichlet allocation," Journal of Machine Learning Research 3, pp. 993-1022, 2003.

[Bo et al. 2009] S.Bo, S. Qiurui, C. Zhong, and F. Zengmei , "A study on automatic web pages categorization," IEEE International Advance Computing Conference pp.1423-1427, 2009.

[Bugzilla 2011] Bugzilla, http://www.bugzilla.org/, 2011.

[Castillo 2004] C. Castillo, "Effective web crawling," (Ph.D. thesis), University of Chile, 2004.

[Cavnar 1994] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," Proceeding of the Symposium on Document Analysis and Information Retrieval, University of Nevada, Las Vegas, 1994.

[Chen et al. 2009] L.Chen, X.Wang, and C.Liu, "An evolving model of software bug reports," International Conference on Information Engineering and Computer Science, ICIECS, pp. 1, 2009.

[Clauset et al. 2009] A. Clauset, C.R. Shalizi, and M.E.J. Newman, "Power-law distributions in empirical data," *SIAM Review* 51(4), 661-703 (2009).

[Cousineasu 2009] D. Cousineau, "Fitting the three-parameter Weibull distribution: Review and evaluation of existing and new methods," IEEE Transactions on Dielectrics and Electrical Insulation, vol.16, no.1, pp. 281-288, 2009.

[Conover 1980] W.J. Conover, Practical Nonparametric Statistics, Second Edition, pp. 465, John Wiley & Sons, Inc, 1980.

[Craven et al. 1998] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K.Nigam, and S. Slattery, "Learning to extract symbolic knowledge from the World Wide Web," In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI- 98), pp. 509-516, 1998.

[Eclipse 2011] Eclipse Source Repository, http://dev.eclipse.org/viewcvs/index.cgi/, 2011.

[Eichmann 1994] D. Eichmann, "The RBSE spider: Balancing effective search against web load," In Proceedings of the First World Wide Web Conference, 1994.

[Ferzund et al.2009] J.Ferzund, S. N.Ahsan, and F. Wotawa, "Software change classification using hunk metrics," IEEE International Conference on Software Maintenance," pp. 471, 2009.

[Garc´ıa et al. 2006] M. Garc´ıa, H. Hidalgo, and E. Ch´avez, "Contextual entropy and text categorization," In Web Congress, LA-Web '06. Fourth Latin American pp.147, 2006.

[Gibbons and Chakraborti 2010] J. D. Gibbons and S. Chakraborti, "Nonparametric statistical inference," Fifth edition, ISBN-10: 9781420077612, ISBN-13: 978-1420077612 2010 Chapman and Hall/CRC, 2010.

[Gittens et al. 2005] M.Gittens, Y. Kim, and D.Godwin,"The vital few versus the trivial many: Examining the Pareto Principle for software," 29th Annual International Computer Software and Applications Conference, pp.179, 2005.

[Guo and Sampath 2008] Y. Guo and S. Sampath, "Web application fault classification-An exploratory study," Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 2008.

[Gupta and Johari 2009] P.Gupta and K. Johari, "Implementation of web crawler," 2nd International Conference on Emerging Trends in Engineering and Technology (ICETET), pp.838-843, 2009.

[Gupta et al. 2006] A. Gupta, O. P. N. Slyngstad, R. Conradi, P. Mohagheghi, H. Ronneberg, and E. Landre, "An empirical study of software changes in Statoil ASA - origin, priority level and relation to component size," International Conference on Software Engineering Advances, pp.12, 2006.

[Hao et al. 2009] P. Hao, D. Ying, and T. Longyuan, “Application for web text categorization based on support vector machine,” International Forum on Computer Science-Technology and Applications (IFCSTA), vol.2, pp.42-45, 2009.

[Hassan 2008] A.E. Hassan, “The road ahead for mining software repositories,” Frontiers of Software Maintenance, pp.48-7, 2008.

[Holts et al. 2010] A. Holts, C.Riquelme, and R. Alfaro,"Automated text binary classification using machine learning approach," XXIX International Conference of the Chilean Computer Science Society (SCCC), 2010, pp. 212-217, 2010.

[Ichii et al. 2008] M.Ichii, M.Matsushita, and K.Inoue, “An exploration of power-law in use-relation of Java Software Systems,”19th Australian Conference on Software Engineering, 2008.

[IEEE 2009] IEEE Std. 1044-2009, “IEEE Standard for classification for software anomalies,” IEEE Computer Society, Software and System Engineering Standards Committee, 2009.

[Java.net 2011] Java.net, The source of Java Technology Collaboration, http://www.java.net, 2011.

[Jira 2011] Jira, http://www.atlassian.com/software/jira/, 2011.

[Joachims 1997] T. Joachims, “A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization,” In Machine Learning: Proceedings of the Fourteenth International Conference (ICML), pp. 143-151, 1997.

[Kim et al. 2008] S. Kim, E.J. Whitehead, and Y. Zhang, "Classifying software changes: Clean or buggy?" IEEE Transactions on Software Engineering, vol. 34, no. 2, pp. 181, 2008.

[Kothari et al. 2006] J. Kothari, T.Denton, A.Shokoufandeh, S.Mancoridis, and A.E.Hassan, "Studying the evolution of software systems using change clusters,” 14th IEEE International Conference on Program Comprehension, pp. 46-55, 2006. 2006.

[Lewis 1991] D.D. Lewis, “Evaluating text categorization,” In Proceeding of Speech and Natural Language Workshop, pp. 312-318, 1991.

[Li et al. 2006] Z. Li, L. Tan, X. Wang, S. Lu, Y. Zhou, and C. Zhai, “Have things changed now? – An empirical study of bug characteristics in Modern Open Source Software,” ACM/Architectural Support for Programming Languages and Operating Systems, pp. 25-33, 2006.

[Matlab 2011] MATLAB, http://www.mathworks.com/, 2011.

[McCallum 2002] A. K. McCallum, "MALLET: A Machine Learning for Language Toolkit," University of Massachusetts, Amherst, Massachusetts, http://mallet.cs.umass.edu/, 2002.

[McCallum et al. 1998] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng, "Improving text classification by shrinkage in a hierarchy of classes," In Machine Learning: Proceedings of the Fifteenth International Conference (ICML), pp. 359-367, 1998.

[Mercurial 2011] Mercurial, http://mercurial.selenic.com/, 2011.

[Minitab 2011] Minitab, http://www.minitab.com, 2011.

[Mozilla 2005] Mozilla, https://bugzilla.mozilla.org, 2005.

[MYSQL 2010] http://www.mysql.com, 2010.

[NetBeans 2011] NetBeans, http://www.netbeans.org, 2011.

[Nigam et al. 1999] K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," In IJCAI-99 Workshop on Machine Learning for Information Filtering pp. 61-67, 1999.

[Pareto 1906] V. Pareto, Manual of Political Economy, pp.13, 1906.

[Paul 2010] Text Classification Tools Version 0.11, W. Paul , Department of Computer and Information Science Temple University, Philadelphia, PA,USA, http://www.cis.temple.edu/~wolfgang/, 2010.

[Pei et al. 2007] J. Pei, J.Han, H.Lu, S. Nishio, S. Tang, and D. Yang, "H-Mine: Fast and space-preserving frequent pattern mining in large databases," IIE Transactions, vol.39, pp. 593-605, 2007.

[Pressman 2010] R.S. Pressman,"Software Engineering: A Practitioner's Approach", 7th edition, Mc Graw-Hill Companies, Inc., ISBN-9780071267823, 2010.

[Pulijala and Gauch 2004] A.K. Pulijala and S.Gauch. "Hierarchical text classification," http://citeseer.uark.edu/publications/CITSA2004.pdf , 2004.

[Qin 2006] Z. Qin, "Naive Bayes classification given probability estimation trees," Fifth International Conference on Machine Learning and Applications (ICMLA), pp. 34-42, 2006.

[Rahmoun and Elberrichi 2007] A. Rahmoun and Z. Elberrichi, "Experimenting N-Grams in text categorization," The International Arab Journal of Information Technology, vol. 4, no.4, 2007.

[SDN 2010] ORACLE Sun Developer Network (SDN), http://bugs.sun.com/bugdatabase, 2010.

[Slonim and Tishby 2001] N.Slonim and N.Tishby, "The power of word clusters for text classification," In 23rd European Colloquium on Information Retrieval Research (ECIR), 2001.

[Thorsten 1997] J. Thorstern, "A probabilistic analysis of the Rocchio Algorithm with TFIDF for text categorization," Fourteenth International Conference on Machine Learning (ICML 1997), pp.143-151, 1997.

[Thorsten 2010] J. Thorsten, "SVM: Support Vector Machine," http://svmlight.joachims.org/, Cornell University, Department of Computer Science, 2010.

[Triola 2009] M.F.Triola, "Elementary Statistics 11th Edition," section 4-8 CD-ROM, 2009.

[Viger 2011], P.F Viger, http://www.philippe-fournier-viger.com/spmf/index.php ?link=algorithms.php, 2011.

[Wajeed and Adilakshmi 2009] M.A. Wajeed and T.Adilakshmi, "Text classification using machine learning," In Journal of Theoretical and Applied Information Technology, JATIT,www.jatit.org, pp. 119-123, 2009.

[Weibull.com 2011] Reliability Engineering Resources, http://www.weibull.com/basics/lifedata.htm, 2011.

[Ying et al. 2004] A.T.T.Ying, G.C.Murph, R.Ng, and M.C.Chu-carroll, "Predicting source code changes by mining change history," IEEE Transactions on Software Engineering, vol.30, no.9, pp.574,2004.

[Zhang 2008] H. Zhang, "On the distribution of software faults," IEEE Transaction on Software Engineering, vol. 34, no 2, pp. 301, 2008.

[Zimmermann et al. 2004] T. Zimmermann, P.Weibgerber, S.Diehl, A. Zeller, "Mining version histories to guide software changes," 26th International Conference on Software Engineering (ICSE'04), 2004.

[Zimmermann et al. 2007] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," Proc. Third Int'l Workshop Predictor Models in Software Eng., http://www.st.cs.uni-sb.de/softevo/, 2007.