A DEFECT PREDICTION METHOD FOR SOFTWARE VERSIONING

by

Yomi Kastro

B.S., Computer Engineering, Boğaziçi University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2006

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor and mentor Dr. Ayşe Bener for her support, lessons and encouragement for the last five years. Her guidance is one of the major pillars throughout this research.

I am also grateful to Prof. Levent Akın for his teachings in the field of Artificial Intelligence.

I would also like to mention the members of Software Engineering Research Lab for their discussions and contributions to my research. And specially to, Onur Kutlubay, Bilge Başkeleş and Burak Turhan.

Besides, I want to thank Pfizer, Business Technologies Division for their endless support and fair indulgence during my intense research period.

And finally, I am deeply thankful to my family for their encouragement and of course, to my dear Doli for her patience, support and for her love…

ABSTRACT

A DEFECT PREDICTION METHOD FOR SOFTWARE VERSION DIFFERENCES

Software lifecycle is becoming more human-independent with the help of new methodologies and tools. Many of the research in this field focus on defect reduction, defect identification and defect prediction. Defect prediction is a relatively new research area using various methods from artificial intelligence to data mining. Currently, software engineering literature still does not have a complete defect prediction solution for new versions of a software product.

In this research our aim is to propose a model for predicting the number of defects in a new version of a software product relative to the previous version by considering the changes. These changes might be introduced as a new feature or a change of algorithm or even as a form of a bug fix. Analyzing the types of changes in an objective and formal manner and considering the lines of code change, we aim to predict the new defects introduced into the new version.

Using such a proposed model will benefit to a more focused testing phase which will decrease the overall effort and cost. Also, this method can help to determine the stability of a software version before publishing the product. The method also helps us to understand the individual effect of a feature, bug fix or change in terms of probability of a new defect introduction.

ÖZET

YAZILIM SÜRÜM FARKLILIKLARI İÇİN BİR HATA TAHMİN YÖNTEMİ

Yazılım oluşturma süreci yeni araçların ve tekniklerin yardımıyla gitgide daha insandan bağımsız bir hale gelmektedir. Bu alandaki araştırmaların önemli bir kısmı yazılım hatalarının azaltılması, hataların tespiti ve tahmini üzerine odaklanmıştır. Yapay zeka tekniklerine ve veri madenciliğine dayalı hata tahmini bu alanda göreceli olarak yeni bir araştırma konusu sayılabilir. Bugün itibariyle, yazılım mühendisliği yazınında halen bir yazılım ürününün yeni versiyonlarına yönelik bir hata tahmin çözümü bütünüyle yer almamaktadır.

Bu araştırmanın amacı, bir yazılımın yeni sürümlerindeki hata oranını eski sürümlerine göre olan değişikliklerini baz alarak tahmin eden bir model ortaya koymaktır. Bu değişiklikler yazılımdaki bir yenilik, bir algoritma değişikliği ve hatta bir hata ayıklama değişikliği olabilir. Bu tür değişikliklerin türünü formal ve nesnel bir bakış açısıyla analiz ederek, ve buna yazılımın hacimsel değişikliğini de katarak, yeni versiyondaki hata oranını doğru bir şekilde tahmin edebilmeyi amaçlıyoruz.

Bu araştırmada önerilen modeli kullanarak, yazılım hayat döngüsündeki test sürecini kısaltabilmek ve harcanan eforu azaltabilmek mümkündür. Buna ek olarak, yeni bir yazılım sürümünün sağlamlığını saptamak bu model sayesinde olasıdır. Bu model, aynı zamanda bir yazılım ürününe katılan yeniliklerin, hata ayıklama değişiklikleri gibi değişiklik türlerinin hata oluşturma ihtimallerine olan katkısını ayrı ayrı anlamaya yardımcı olmaktadır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES viii
LIST OF TABLES x
LIST OF ABBREVIATIONS xi
1. INTRODUCTION 1
1.1. Motivation 1
1.2. Outline 2
2. BACKGROUND 3
2.1. Software Defect Prediction Models 3
2.1.1. Predicting Defects Using Bayesian Belief Networks (BBNs) 3
2.1.1.1. An Overview of BBN in Software Defect Detection 4
2.1.1.2. The BBN model in Defect Detection 5
2.1.2. Software Metrics 8
2.1.3. Using Machine Learning and DT for Defect Density Estimation 9
2.1.3.1. Results and Evaluation of Model 10
2.1.4. Defect Identification Using Machine Learning Techniques 11
2.1.4.1. The AI Point of View of the Model 12
2.2. An Overview of Software Defect Detection Literature 13
3. PROBLEM STATEMENT 16
3.1. Benefits of Defect Prediction 16
3.2. Version Change Tracking 18
3.3. Sample Demonstration of the Problem 19
4. PROPOSED MODEL 21
4.1. Defining and Tracking Change 21
4.2. Defect Identification Over Versions 22
4.3. Merging Change and Defects 22
4.4. Building the F function 24
4.5. Proposed Defect Prediction Framework 25

5. DATA COLLECTION PROCESS 27
5.1. Data Requirement Analysis 27
5.2. Data extraction on change analysis 30
5.3. Data extraction on CVS logs 39
6. EXPERIMENTS 42
6.1. Using Neural Networks 42
6.2. Deciding Hidden Layers and Units 42
6.3. Building the Neural Network 43
6.4. First Data Set 44
6.4.1. Choosing the Correct Learning Algorithm 44
6.4.2. Deciding Learning Rate 45
6.4.3. Epoch vs. Error Analysis 46
6.4.4. Performance Results 47
6.5. Second Data Set 51
6.5.1. Epoch vs. Error Analysis 51
6.5.2. Performance Results 52
7. CONCLUSION 58
7.1. Contributions 58
7.2. Future Work 59
8. APPENDIX A: SOURCE CODE 60
8.1. MLP Driver 60
8.2. MLP Base Function 63
9. APPENDIX B: DATA 67
9.1. Azureus Change Data 67
9.2. Linux Kernel Data 102
REFERENCES 103

LIST OF FIGURES

Figure 2.1 “Reliability prediction” BBN example of Norman Fenton..... 4

Figure 2.2 Node Probability Table for the Node “Reliability” by Norman Fenton . 5

Figure 2.3 The model of BBN for defect detection by Norman Fenton 6

Figure 2.4 A demonstration of Goldilock’s Conjecture by Norman Fenton..... 7

Figure 2.5 The predicted values and the real values on an example 11

Figure 2.6 The predicted values and the real values on an example run..... 11

Figure 2.7: Learning System Architecture by Ceylan, Kutlubay and Bener..... 12

Figure 4.1 Proposed Defect Prediction Framework 26

Figure 5.1 Azureus project details..... 31

Figure 5.2 Interface of bug tracking software in Azureus project..... 32

Figure 5.3 Versions vs. bug occurrence 34

Figure 5.4 Versions vs. sum of occurrence with bugs’ priorities..... 35

Figure 5.5 Weighted defect densities vs. versions 36

Figure 5.6 Change type distributions vs. versions..... 38

Figure 5.7 The CVS log analysis of kernel files between versions..... 40

Figure 5.8 Linux bug searching panel 41

Figure 6.1: Neural network model used for evaluating dataset..... 44

Figure 6.3 Error vs. epoch for different learning rates 45

Figure 6.4: Error vs. epoch rate for 1000 epochs 46

Figure 6.5: Error vs. epoch, detailed 47

Figure 6.6. Error vs. versions for different executions..... 48

Figure 6.7: Real Values vs. Predicted Values for First Dataset 49

Figure 6.8: Error vs. Epochs for second dataset..... 51

Figure 6.9: Error vs. Epochs for second dataset, detailed 52

Figure 6.11: Real Values vs. Predicted Values for Second Dataset..... 53

Figure 6.12: Real Values vs. Predicted Values for Second Dataset, detailed 54

Figure 6.13: Real Values vs. Predicted Values for Second Dataset, Zoom In..... 54

LIST OF TABLES

Table 5.1 Example of Data Format Extracted for Analysis 30
Table 5.2 Stable versions and launch data table..... 33
Table 5.3 Example dataset of change over versions 37
Table 5.4: Final dataset containing both change and defect information..... 39
Table 6.1: Real Values vs. Predicted Values for First Dataset..... 49
Table 6.2: Real Values vs. Predicted Values for Second Dataset 55

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
BBN	Bayesian Belief Network
CMM	Capability Maturity Model for Software
CMMI	Capability Maturity Model Integration
COCOMO	Constructive Cost Model
CVS	Concurrent Versions System
DT	Decision Tree
LOC	Lines of Code
MLP	Multi Layer Perceptron
PCA	Principal Component Analysis
RBF	Radial Basis Functions

1. INTRODUCTION

1.1. Motivation

Software development is still a human-driven process. Although there are various formal methods and tools to improve the automation of a software delivery, the major part is human-dependent. The errors are mostly introduced because of the human involvement in the process. In the foreseeable future, the human interaction to software development processes will continue. Thus, there are proposed methods to make software development life cycle independent of human-abilities by basing them on well-defined processes [1,2,3,4,5] .

Much research effort is given in the field of software quality [1,3,6,7]. These efforts include formal methods building or offering new methodologies like Capability Maturity Model (CMM) or even producing tools for increasing quality.

Software quality is directly correlated with number of defects in the software product. So, defect prediction is an important subject of software quality literature. Our research is motivated by the aim of reducing testing efforts and the cost of producing high quality software.

In this research we provide a language and platform independent software defect prediction model where our primary focus is on multi-version software. We are observing the changes between versions to propose a defect density metric for deciding a new stable version.

1.2. Outline

In the next chapter, background information is given about the topics that are related to our work presented in this research. First, artificial intelligence application in software quality work is discussed. Then, general software quality approaches and current research are briefly explained.

Second chapter also is a brief literature survey in this field. Defect prediction and defect identification focused research is also discussed in this chapter.

In 3, the main goal and the problem statement of this research is presented. Some challenging issues throughout this research are also mentioned.

In Chapter 4, we present our proposed model and solution. The success conditions of the proposed method are extensively discussed.

In Chapter 5, we propose data collection methods and technique required for this research. Data collection was a major timeline item for this research. Data validity and objectivity are two main discussions in this chapter.

In Chapter 6, we discuss the experiments structure and we evaluate the experiment results. A comparison of two experiments is done.

In Chapter 7, we provide the evaluation of the method and explain some future research areas. A summary of the research is done and contributions are explained to conclude this thesis.

2. BACKGROUND

In this Chapter we provide brief overview of various methods to increase software quality and specifically decreasing the defect density. We discuss artificial intelligence techniques that are used for defect prediction and identification. These are Bayesian Belief Networks, Neural Networks and Decision Trees.

We first provide an overview of artificial intelligence applications on software quality and then we will give a brief discussion on other defect prevention methods.

2.1. Software Defect Prediction Models

In software quality literature one of the most discussed problems is software defect prediction. This concept has been evaluated as both prediction and the estimations of the defects in a given software module. Generally, many efforts are concentrated specifically in predicting the number of defects in the system, estimating the reliability of the systems as statistical functions to time-to-failure, and understanding the importance of design and testing processes on defect counts.

One of the most common examples of metrics is the lines of code. There are many previous works using lines of code as at least one of the metrics [2,3,6,9]. We give a more detailed discussion in Section 2.1.2 of this chapter.

2.1.1. Predicting Defects Using Bayesian Belief Networks (BBNs)

It is mentioned that trying to detect defects using only size metrics is not accurate [9,13]. The number of defects discovered is clearly related to the amount of testing performed. It is obvious that a program which has not been tested, or used, will have a zero defect count; even the program is very complex. There are approaches that modeling the complexities of software development using new probabilistic techniques presents a positive way forward [1]. These are Bayesian Belief Networks (BBNs). Using BBNs we are able to express complex interrelations within the model at a level of uncertainty commensurate with the problem [1].

2.1.1.1.An Overview of BBN Usage in Software Defect Detection

Bayesian Belief Networks have different naming in the literature. These are Belief Networks, Causal Probabilistic Networks, Causal Nets, Graphical Probability Networks, Probabilistic Cause-Effect Models, and Probabilistic Influence Diagrams. BBNs are mainly used in uncertain decision support cases [1]. Recent algorithms allowed us to realize BBNs nowadays although the mathematical background is available for long time [1].

A BBN is a graphical network that represents probabilistic relationships among variables. It is possible to conclude reasoning under uncertainty and combine the advantages of an intuitive visual representation with a sound mathematical basis in Bayesian probability. It is possible to articulate expert beliefs about the dependencies between different variables and to propagate consistently the impact of evidence on the probabilities of uncertain outcomes, such as "future system reliability." Actually BBNs reflect expert opinions in each individual node. Figure 2.1 shows a BBN for an example "reliability prediction" problem by Norman Fenton [1]

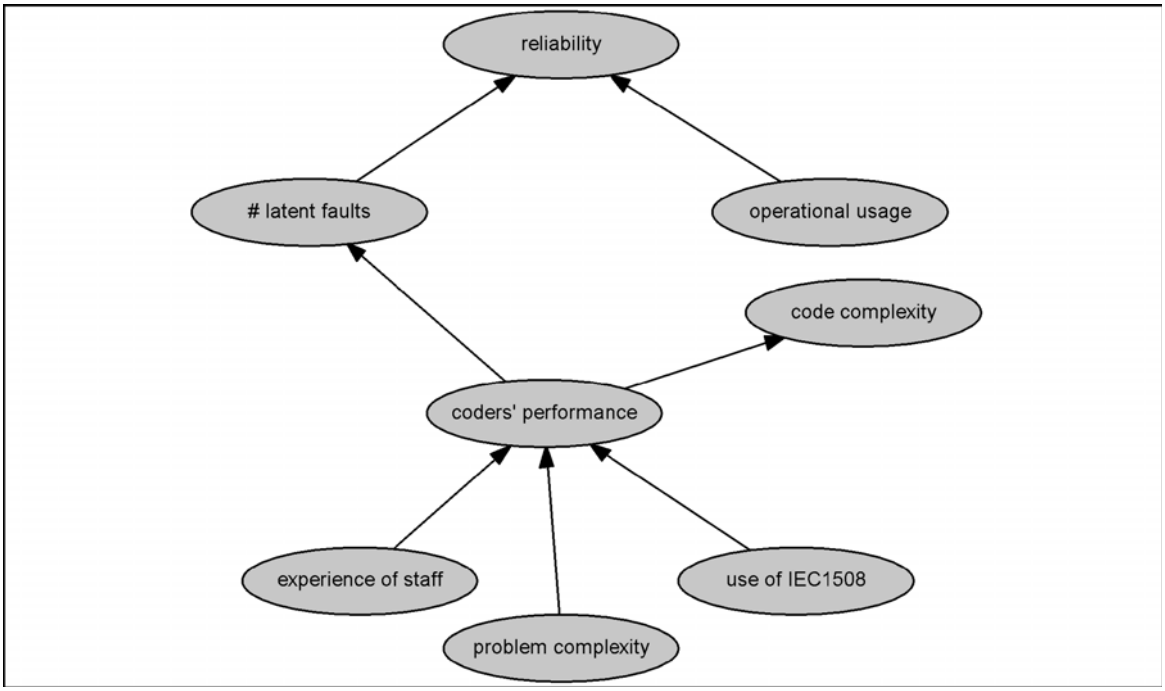


Figure 2.1 "Reliability prediction" BBN example of Norman Fenton [1]

In the figure discrete or continuous variables are represented by nodes. The arcs represent causal/influential relationships between variables. For example, software reliability is defined by the number of faults. In Figure 2.2 by Norman Fenton [1] for simplicity all nodes are discrete so that here reliability takes on just three discrete values as low, medium, and high.

operational usage		*low*			*med*				*high*	
faults		*low*	*med*	*high*	*low*	*med*	*high*	*low*	*med*	*high*
reliability	*low*	0.10	0.20	0.33	0.20	0.33	0.50	0.20	0.33	0.70
	med	0.20	0.30	0.33	0.30	0.33	0.30	0.30	0.33	0.20
	high	0.70	0.50	0.33	0.50	0.33	0.20	0.50	0.33	0.10

Figure 2.2 Node Probability Table for the Node “Reliability” by Norman Fenton [1]

In the industry, especially on the SERENE project which is a decision support system, very large BBNs are built, with very large probability tables [1]. There are many advantages of using BBNs, the most important one is the ability to represent and manipulate complex models that might never be implemented using conventional methods. Also it is possible to predict events based on partial or uncertain data [1].

The major difficulty depends on the individual trying to understand and describe the nature of the problem as well as the problem itself [1]. There might be problems which may appear difficult to a novice programmer but not to an expert [1].

2.1.1.2.The BBN model in Defect Detection

Norman Fenton developed a model for execution and development of BBN [1]. It shows the possibility of combining the different software engineering schools of thought on defect prediction into a single model. Using this model we are able to show how predictions might be made and explain historical results more clearly.

The majority of the nodes have the following states: “very-high,” “high,” “medium”, “low”, “very low.” For the sake of completeness, the probabilities attached to each of these states are fictitious but are determined from an analysis of the literature or common-sense

assumptions [1]. There are two stages of defection detection using BBN. First stage is the design, development phase and the second stage includes testing. In Figure 2.3 the complexity of the problem shows the degree of complexity [1].

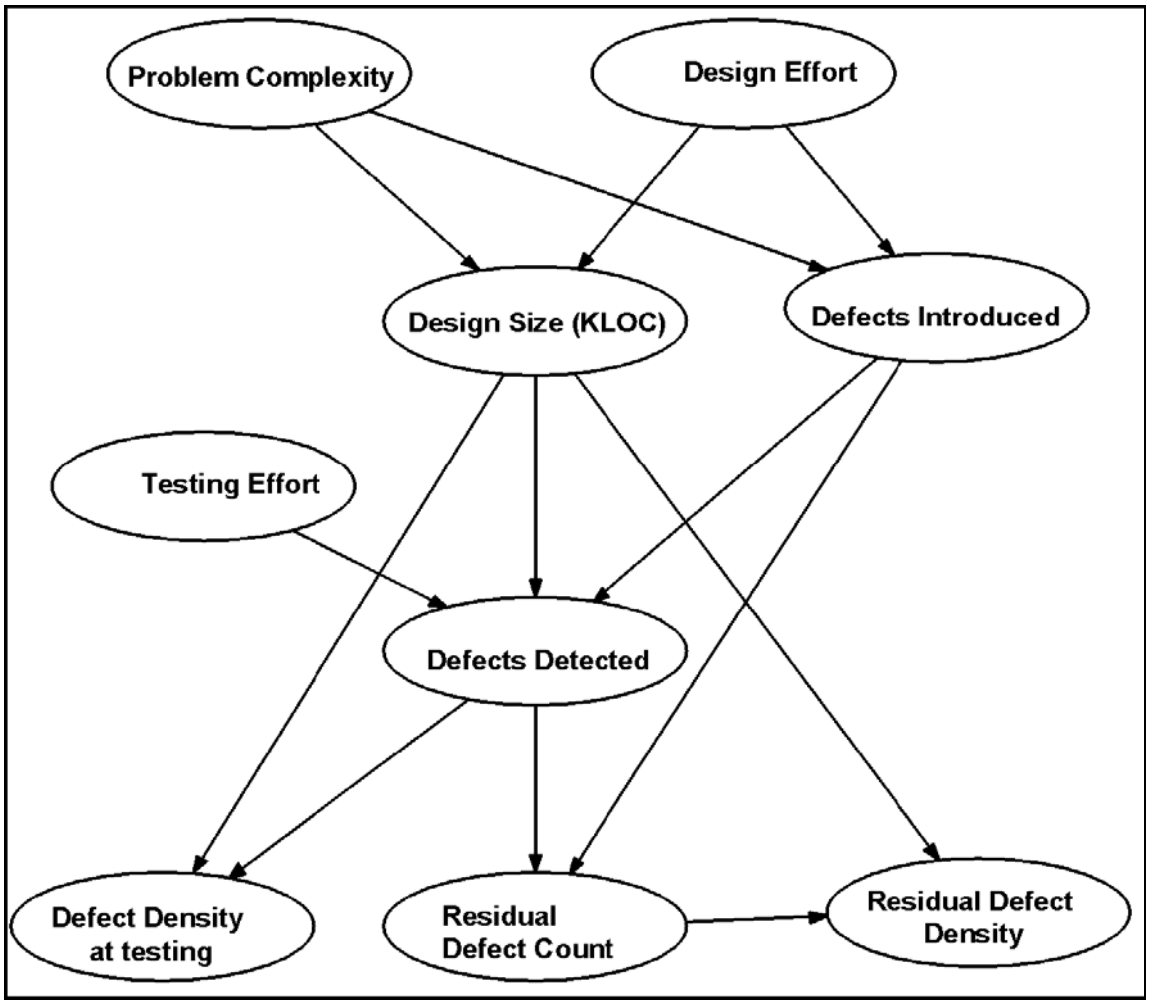


Figure 2.3 The model of BBN for defect detection by Norman Fenton [1]

The testing stage follows the design stage and in practice the testing effort allocated may be much less than that is actually required [1]. The mismatch between testing effort and design size will influence the number of defects detected, which is bounded by the number of defects introduced. Residual defects count is used to find the difference between the defects detected and defects introduced [1]. Norman Fenton claims that *"The defect density at testing is a function of the design size and defects detected (defects/size). Similarly, the residual defect density is residual defects divided by design size."* [1]

The research also mentions the “Goldilock’s Conjecture” which is the idea that there is an optimum module size that is “not too big nor too small.”[1]. In Figure 2.4 it is shown that the execution of the defect density BBN model under the “Goldilock’s Conjecture” [1]. Each window represents a node with the statistical analysis of the predictions made based on the facts entered.

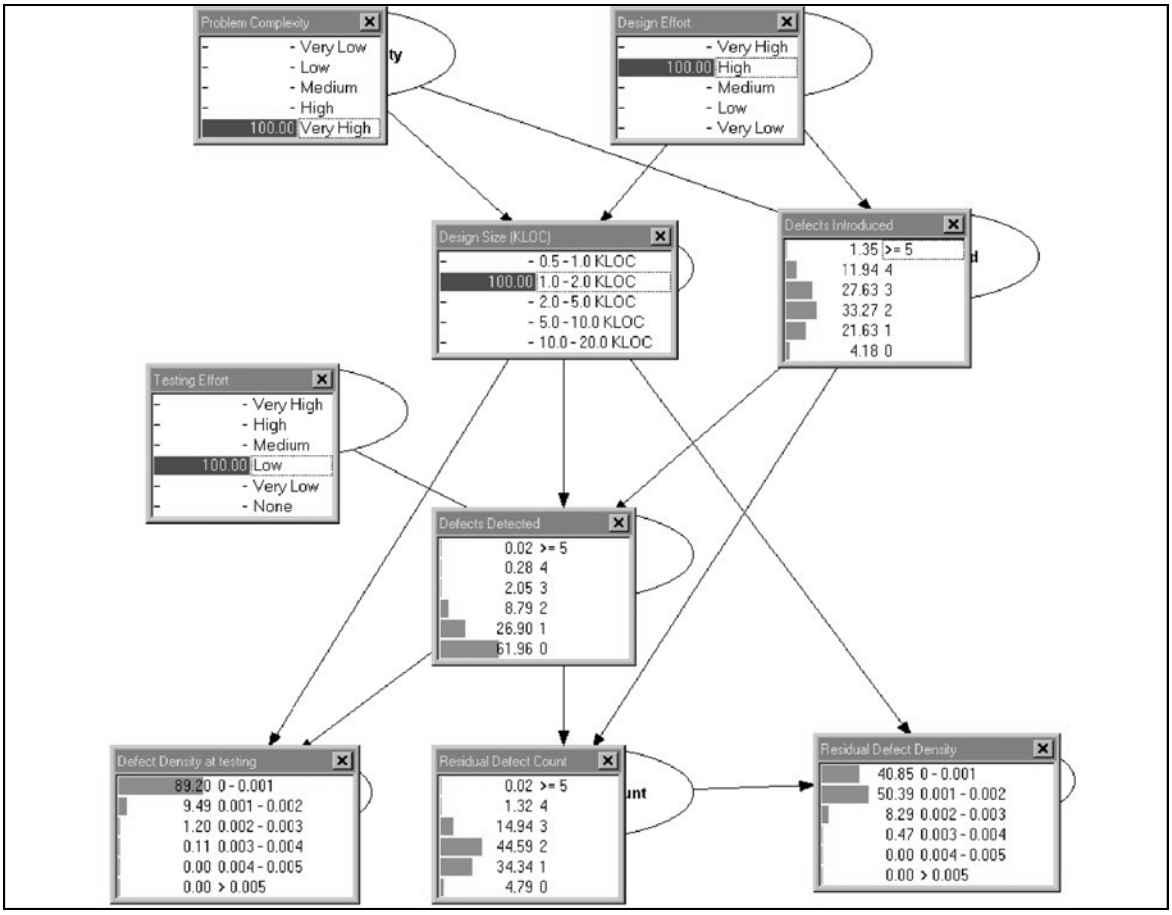


Figure 2.4 A demonstration of Goldilock’s Conjecture by Norman Fenton [1]

Norman Fenton claims that:

“ A very complex problem is represented as a fact set at “very high” and a “high” amount of design effort is allocated, rather than “very high” commensurate with the problem complexity. The design size is between 1.0–2.0 KLOC. The model then propagates these “facts” and predicts the introduced defects, detected defects and the defect density statistics. The distribution for defects introduced peaks at two with 33 percent probability but, because less testing effort was allocated than required, the distribution of defects

detected peaks around zero with probability 62 percent. The distribution for defect density at testing contrasts sharply with the residual defect density distribution in that the defect density at testing appears very favorable."[1]

Unfortunately, the proposed BBN model is not suitable for defect prediction with version tracking. Because, tracking each defect group for all versions requires a different approach containing historical bugs' data.

2.1.2. Software Metrics

Software metrics are mostly used for the purposes of product quality and process efficiency analysis and risk assessment for software projects. Software metrics have many benefits and one of the most significant benefits is that they provide information for defect prediction. Metric analysis allows project managers to assess software risks.

A software metric is defined as "a quantitative measure of the degree to which a system, component, or process possesses for a given attribute." by Halstead[52]. Based on this definition, choosing the correct software metrics is one of the most important parts of defect prediction.

Software life cycle contains five common phases and the metrics are distributed over these phases. [53]. Some of the well-known metrics and related phases are listed below:

i. *Management* : Cost, schedule, progress, computer resource utilization
ii. *Requirements* : Completeness, traceability, consistency, stability
iii. *Design* : Size, complexity, modularity, coupling, cohesiveness
iv. *Code* : Fault density, problem report analysis, standards compliance
v. *Test* : Coverage, sufficiency, failure rate, mean time to failure

Generally, software size is quantified by size metrics [52]. Lines of code (LOC) is the most widely used and the most easily obtained metric for program size [52,53]. This metric can be extended to "Non-commented LOC" or "LOC with comments" depending on counting the blank and commented lines.

There are also Complexity metrics to measure program complexity. McCabe's measure is the commonly accepted complexity measure [55]. Cyclomatic complexity and extensions to it are widely used in the industry [55].

An important metrics group is *Halstead's metrics. Halstead* proposes a unified set of metrics that apply to several aspects of programs, as well as to the overall software production effort [52]. Sum of the number of unique operators in the program and the number of unique operands in the programs are used as Halstead vocabulary definition. Likewise, program length is calculated by summing the total number of operators in the program and the total number of operands in the program.

The early researches on software metrics have focused their attention mostly on McCabe, Halstead and LOC metrics. Among many software metrics, these three categories contain the most widely used metrics. Also in this research, we decided to use a model mainly based on these metrics.

2.1.3. Using Machine Learning and DT for Defect Density Estimation

In their research Bener and Kutlubay aim to figure out the defective modules using the software metric data as an input [2]. The code metric data is taken from NASA's Metrics Data Program (MDP). The repository contains software metric data and error data at the function/ method level. Kutlubay and Bener constructed a two-step model that predicts potentially faulty modules by taking their metric data into consideration. This model uses classification and regression type learning methods consecutively. The results of the experiments show that the two-step model enhances the regression performance.

Some of the metrics data used are McCabe Metrics; Cyclomatic Complexity and Design Complexity, Halstead Metrics; Halstead Content, Halstead Difficulty, Halstead Effort, Halstead Error Estimate, Halstead Length, Halstead Level, Halstead Programming Time and Halstead Volume, Lines of Code (LOC) Metrics; Lines of Total Code, LOC Blank, Branch Count, LOC Comments, Number of Operands, Number of Unique Operands and Number of Unique Operators, and lastly Defect Metrics; Error Count, and

Error Density-reference. They have divided the data set into two groups; the training set and the testing set. These two sets are randomly generated from the overall data.

2.1.3.1.Results and Evaluation of Model

First attempt of the authors did not bring out successful results, because the shuffle algorithm produced large variances between data sets and MSE values. The experiment shown in Figure 2.5 run using PC4 project. According to this graph, it is clear that the method performs faulty predictions over the modules that are in fact defect-free. The modules which do not contain any actual defects predicted as they contain defects. The points on the y-axis show that there are vast amount of faulty predictions for defect-free values.

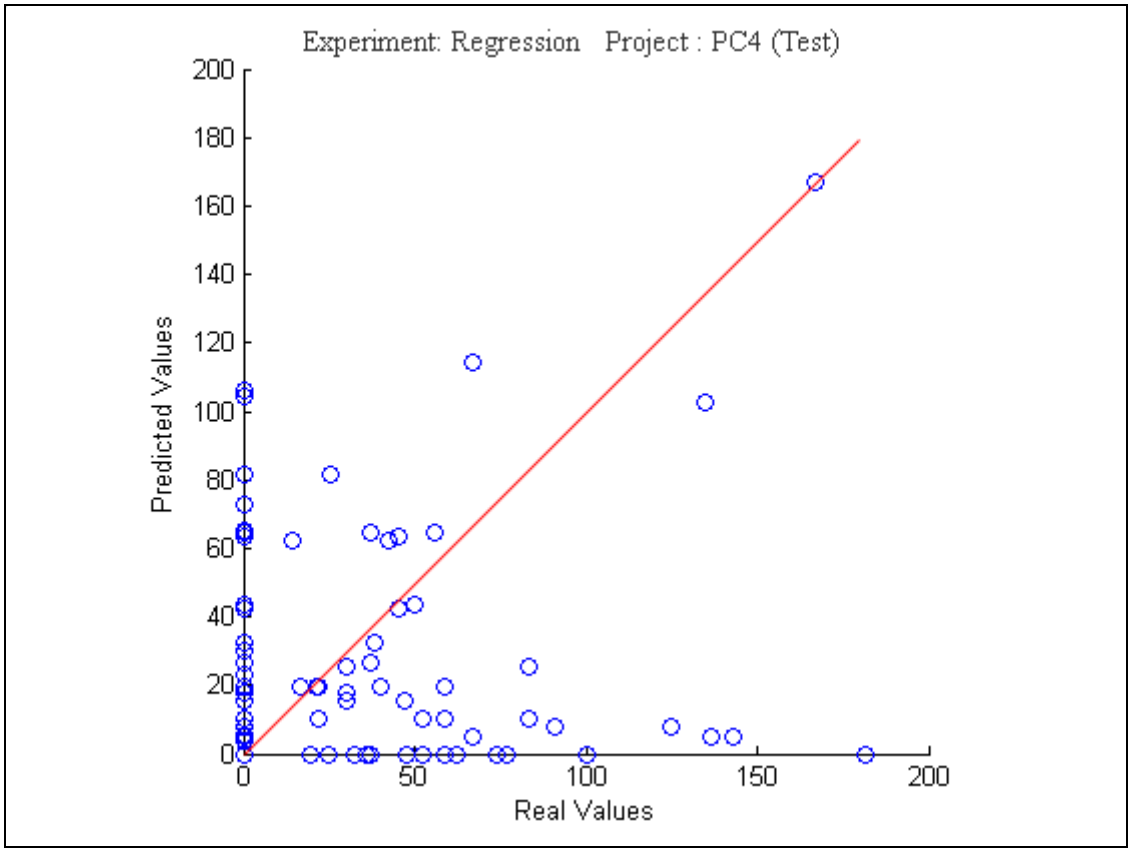


Figure 2.5 The predicted values and the real values on an example run for project PC4 by Bener and Kutlubay [2]

After this unsuccessful experiment, the authors conducted a new experiment with only defected modules. And the result is shown in Figure 2.6.

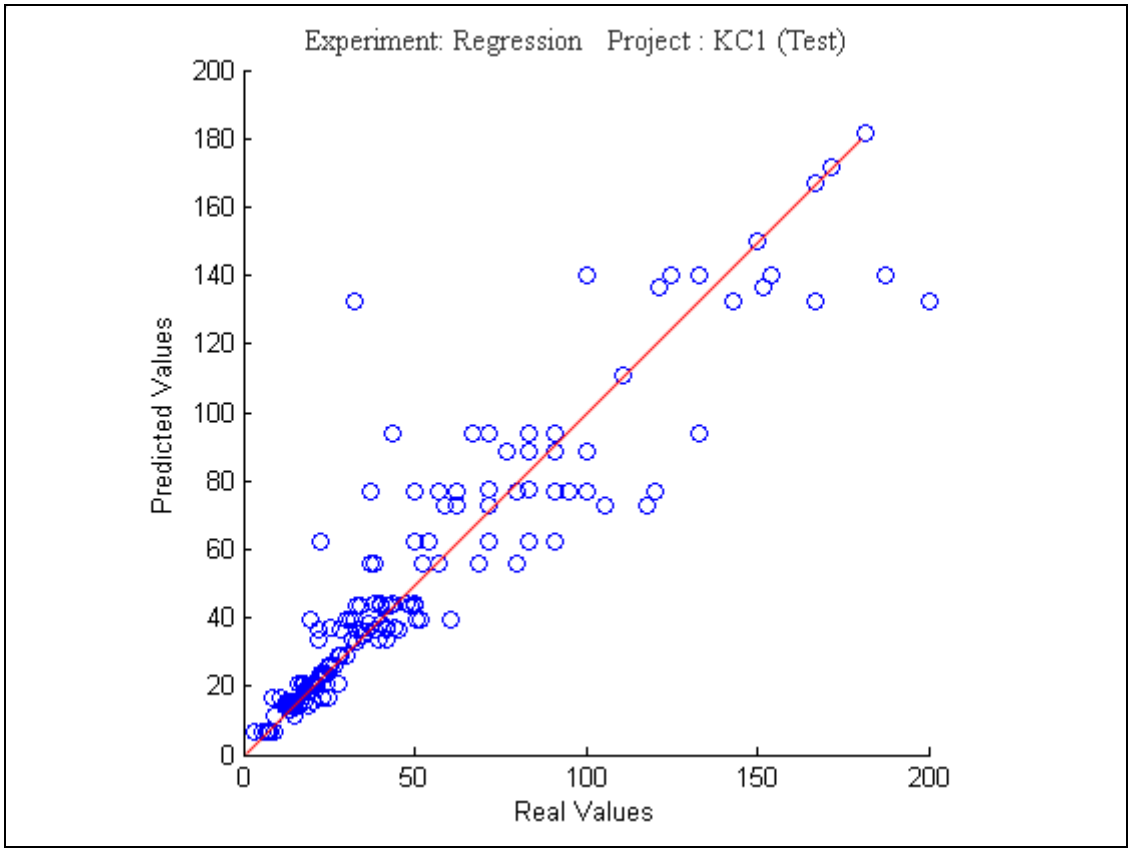


Figure 2.6 The predicted values and the real values on an example run for KC1 project where the input data set contains only defected items by Kutlubay and Bener [2]

The major contribution of this research is, using both classification and regression techniques together for the first time.

2.1.4. Defect Identification Using Machine Learning Techniques

In another research, Bener, Kutlubay and Ceylan proposed a model using machine learning methods [3]. Principal Component Analysis is used for dimensionality reduction, and Decision Tree, Multi Layer Perceptron and Radial Basis Functions are used for defect prediction. The experiments in this research are carried out with different software metric datasets that are obtained from real-life projects of three big software companies in Turkey.

2.1.4.1.The AI Point of View of the Model

The collected input data in each dataset has many correlated values, principal component analysis (PCA) is carried out for transforming the correlated variables into uncorrelated ones. Primarily, PCA is reducing the dimensionality of the datasets by eliminating the correlations among the attributes. After the implementation of PCA, a new dataset is obtained with fewer dimensions.

After producing the resultant dataset, the system is trained with this optimized dataset using the regression based algorithms such as decision tree (DT), multi layer perceptron (MLP) and radial basis functions (RBF). The outputs in the system state the number of total defects per module/ function. Then the system is tested by using the test data which is part of the entire dataset. After the testing process, the desired output is obtained. The output represents the total number of defects per each module/ function. This flow is described in Figure 2.7.

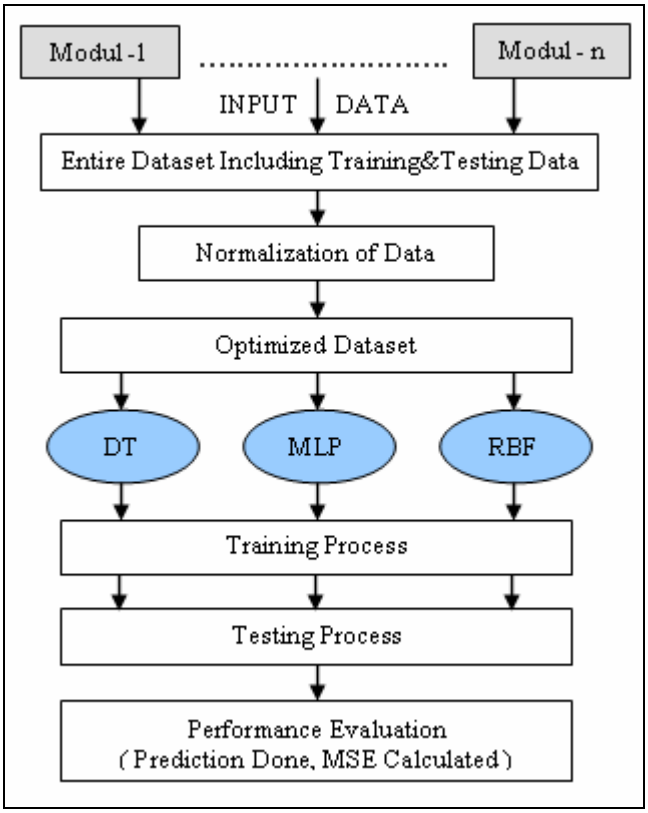


Figure 2.7: Learning System Architecture by Ceylan, Kutlubay and Bener [3]

2.2. An Overview of Software Defect Detection Literature

There is various different research in the field of defect detection. An important statistical research is done by Boehm and Basili [6], which helps us to understand the scale of the problem. The research [6] has found out, finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase. They also claim that about 80 percent of avoidable rework comes from 20 percent of the defects.

Additionally, about 80 percent of the defects come from 20 percent of the modules, and about half the modules are defect free. Interestingly, peer reviews catch 60 percent of the defects. Their eighth [6] issue is disciplined personal practices can reduce defects introduction rates by up to 75 percent. Significantly, about 40 to 50 percent of user programs contain nontrivial defects. This research gives us an important scale factor of the problem.

Some of the defects are carried from the requirements phases to the final products. So, there exists some research focused on inspections of software requirements to detect defects [7]. One of the most significant papers in this area is published by Porter and Votta [7], claiming that a scenario-based detection method, in which each reviewer executes a specific procedure to discover a particular class of defects has a higher defect detection rate. They briefly describe the design, execution and analysis of the experiment on different populations of software engineers. Finally, they claim that scenario based approach is superior to other methods with an improvement of 35% [7].

A major research focused on N-version software experiment is conducted by Brilliant and Knight [8]. A program is written 27 times independently by various universities using the same requirements. They tried to find out if the defects are statistically independent. They have found out that in some cases the programmers made equivalent logical errors, indicating that some parts of the problem were simply difficult than others. Finally they conclude that, minor differences in the software development environment, such as the use of different programming languages for the different

versions, would not have a major impact in reducing the incidence of faults that cause correlated failures [8].

In some research the location of the defects was an issue as well as the total number of them. Ostrand and Veyuker conducted a research in 2005 focusing on both location and number of faults in large software systems [9]. The predictions are based on the code of the file in the current release, and fault and modification history of the file from previous releases. The model has been applied to two large industrial systems, one with a history of 17 consecutive quarterly releases over four years and the other with nine releases over two years. The prediction was very accurate, identifying files that contained from 71 percent to 93 percent of the faults, with the average being 84 percent. The predictions were based on a negative binomial regression model.

Machine learning models are also used is some research to estimate the defect content after an inspections. Padberg, Ragg and Scholnecht [10] tried to make a system learn from empirical data. Specifically, the relationship between certain observable features of an inspection and the number of defects actually contained in the documents. They showed that some features can carry significant nonlinear information about the defect content. Therefore, they used a nonlinear regression technique, neural networks, to solve the learning problem. They validate their data such that their model is successful even with a small dataset. Their approach yielded much more accurate estimates than the standard estimation methods such as capture-recapture and detection profile. They also detected and exploited nonlinear relationships between the features of an inspections and the defect content of the document. They claim that their approach automatically adapted to different document types and sizes, reading techniques, and team sizes.

Besides neural network models, there are also some probabilistic models to predict both number of defects and software development effort. Pendharkar and Subramanian have shown progress in this field [11]. They have used bayesian networks with popular nonparametric networks and regression tree forecasting models.

Some research in this field use code churn measures. A research is sponsored by Microsoft Research and conducted by Nagappan and Ball [12], inspects code churn

measuring the changes made to a component over a period of time, quantifying the extent of this change. They present a technique for early prediction of system defect density using a set of relative code churn measures such as component size and the temporal extent of churn. Statistical regression models are used to show absolute measures of code churn are poor predictors of defect density but the relative measures are performed better. They have reached to an accuracy of 89.0 percent. This research was based on Windows Server 2003 operating system. They simply showed that an increase in relative code churn measures is accompanied by an increase in system defect density. Additionally, using relative values of code churn predictors is better than using absolute values to explain the system defect density. Also, they claim that relative code churn measures can be used to discriminate between fault-prone and not fault-prone binaries.

Tracking change in software was the key of predicting the defect density for a considerable amount of research. Thus, change impact identification was taken into account by Kung, Gao and Hsia [13] in object oriented software scope. They have described a formal model for capturing and inferencing on the changes to identify affected classes is described. Their model including object relation diagram, the block branch diagram and the object state diagram. They were primarily focused on C++ projects. They claim that the changed and affected classes can be tested in a cost-effective order to avoid extensive construction of test stubs.

To conclude with, there has been various research for defect prediction of software products. Some of them use a large number of different metrics as input and some of them focus a strong subset of metrics. Mainly, neural networks and MLP applications are used, where as there exists major contributions with BBNs and decision trees.

3. PROBLEM STATEMENT

In this chapter, we will establish the primary focus of the thesis. A successful defect prediction has various benefits for all parties involved with the final software product. So, we will discuss the clear benefits of a successful solution.

Most of the research in this field is focused on a single software product rather than a series, namely versions of the software. So, we will try to figure out the version concept and the effects for defect prediction models. Using a version-based scheme has some advantages and disadvantages that we will discuss.

Finally, a sample demonstration of the problem will be given in terms of a software development institute or company that can benefit from a successful defect prediction solution.

3.1. Benefits of Defect Prediction

The software can be classified into groups which require zero defect density, for example aeronautic purposes, or low defect density like operating systems and system critical software or the group that can tolerate high levels of defect density like non-critical user applications [11,20].

In zero defect density scenarios, the defected software is impossible for use or to propose as a product. So the software development company is fully focused on finding all defects and fixing them. A defect prediction methodology will be inevitable in such situations because predicting defects and locating them will help us to deliver a defect-free solution.

In the case of operating system software or in the case of non-critical user applications the software is designated to be defected below a certain defect density which is set depending on the application. At this point, a relation is put along with the defect density and time-to-failure. Using this relation, the software developer institution can target a defect density level and arrange a testing and correction phase to reach that defect

level. In most of the cases, formal review methodologies for requirements and design are executed as well as the testing process of the software [24,25,26]. Testing is scheduled as unit testing and functional testing, but also regression testing against change is done. Testing effort is a major part of the overall project effort and cost. There are various research on how to distribute this effort in order to succeed a high quality software [11,19,20].

Presently, a defect prediction solution helps the software developer institution to distribute their testing resources in parallel with the defect density. We can argue that the modules with higher defect density should have a higher share than regular distribution, whereas the modules with lower defect density should have a lower share than the regular. A version-based defect density solution may also consider the changes. If the software institution would supply the change information quantitatively, the defect prediction solution can predict in relation to these changes. In this case the regression testing effort may also be reduced.

A defect prediction solution might also provide a guideline to the sources of defects. These defects might be caused due to programmer's inability, failure in requirements collection or design mistakes. Thus, a defect prediction model with source identification can give important ideas regarding the erroneous bottlenecks in the software development cycle. Especially, efficiency focused software development units can benefit using defect cause information. They can take necessary precautions in a proactive manner. Briefly, a defect focused prediction solution can also help to change the development methods. Such a solution or systematic approach can affect in a positive manner to produce less defected software.

Another important aspect of a defect prediction solution is that such a solution becomes necessary when there is a trade-off between to deliver earlier and to deliver with fewer defects. In today's software development industry, all companies and software development houses are in a severe competition that minimizing development time decreases the overall project cost [11,19]. On the other hand, less development and testing time also increases the defect density ratio in the final product. So, with this fact the executive management of the software company should require a quantitative indicator to

find the correct point in this balance. Therefore a defect prediction solution may provide the required quantitative metric to make a decision on the product delivery. The senior management of the software development company would be able to decide launching the product if the defect density level is below a certain threshold.

3.2. Version Change Tracking

Software products are evolving and getting better [30,39]. The same product with the same name with some features added or new functionalities introduced is announced as a new version. A new version might differ from an older version in various ways. The most common differences are newly introduced features due to new requirements, changes of previous modules or bug fixing of the old version. Although most of the software houses claim that the stable version of their software does not contain bugs, we all come across with the fixes in their new versions [25].

The new version of the software most probably will also differ from the older version in terms of lines of code and Code Versioning Systems (CVS) logs. So, the developer institution mostly has the quantified data to measure the change between versions [20].

More importantly, every software product has its own dynamics. Depending on its programming language, programmers' experience, algorithm complexity or even development methodology makes software unique. Thus, every software product should be evaluated with its own dynamics in mind. The best example to reflect a new software dynamics is the previous versions of that specific software. If the developer institution is able to track the bugs of previously published version and the changes among them, these data might be analyzed with a defect prediction model.

To sum up, using previous versions of software to predict its own behavior on reacting to change is expected to give high accuracy [8].

In these conditions, if a version changes tracking system is used in a defect predictor solution; this solution is not applicable to brand new software. It will be rather suitable for new versions of existing software [8].

3.3. Sample Demonstration of the Problem

Let us assume that a software developer company XYZ produced operating system software for servers. They have launched several versions and sub versions until now. Since they have a good documentation policy they have recorded every single change between versions in terms of change types per each module. For each version and each module they have recorded number of features introduced, number of changes imposed and number of bug fixes accomplished to fix previous version bugs.

In addition to that company has also collected the customer data about their bugs of each version. They are aware of total number of bugs of every stable version that they have launched, so that they will be able to fix those bugs for the next version.

Their competitor is also about to launch their competitive product. So the executives of XYZ should make a critical decision about their launch time of the product. They have nearly finalized the development but there exists a testing and a debugging phase in order to have stable software. At this point, they look for a solution which can give their product specific analysis according to previous data and experience to estimate the defect content at any instance. If they would be able to quantify the defect level, a better decision would be made.

In this case, a defect prediction solution is inevitable from the software developer company's point of view. Such a solution will be invaluable. The company can both see the estimate of overall defect density and the probable distribution of defects. Also, they would be able to allocate their software testing workforce more efficiently.

More importantly, they will be able to realize the most defect introducing actions. For example, they might be introducing new bugs mostly in new feature introduction or they are increasing the bugs while trying to fix the existing bugs. If they have such knowledge, they would have the ability to train their developers or change their methodology. In this case, they will be minimizing the effects of actions that have high

defect probability rate. The defect prediction system would be used in a proactive prevention strategy rather than trying to clean them up after they are produced.

4. PROPOSED MODEL

Our aim is to understand the effect of change to versions of software products. In this chapter, we will be proposing a definition for change in the first section. There are two different change identification methods.

In the second section, we will be proposing our defect identification model for currently delivered software products and a method to maintain the high quality of data sources. In the final section we will be proposing a method to build up a correlation among these data and a prediction method.

4.1. Defining and Tracking Change

Change is the core input of any prediction algorithm or method. We want to give a definition of change in our software versioning scope. We call “a change exists” when a line from the source code is modified, deleted or moved. In this scope, change can be easily analyzed by a CVS program. If the software versions are being kept in a software CVS repository two versions can be compared easily in terms of line changes. This definition of change is objective and cannot change from developer to developer.

On the other, we can call “a change exists”, if the developer of any module can specify the minimum modification that produces a feature or bug fix. Thus, finding the lowest level of change is human-dependent thus all versions’ change list should be prepared by the same team or person to overcome subjectivity. This assigned person or team should classify changes into groups of features, bug fixes and other changes. The definition and examples of each group is explained in chapter 5 in more detail.

At the end of change collection procedure, for each module we should come up with either:

i. Number of lines of source code changed
ii. Number of lines of source code inserted
iii. Number of lines of source code deleted

Or

i. Number of features introduced
ii. Number of bug fixes accomplished
iii. Number of modifications done

Another important metric of change is the size change of software, so each version's size difference should also be collected.

4.2. Defect Identification Over Versions

The term "defect" and the term "bug" were used interchangeably throughout this thesis. Thus, bug tracking software is the major defect identification resources for a defect prediction scheme. During the life cycle of any version of a software product, the bug reports and descriptions should be collected extensively. The module information and the criticality of the bugs should also be specified. It is important that the bug reporting entries are moderated and a team or person should approve listing and criticality levels of the bugs to avoid subjectivity.

Most of the bug tracking software has built-in statistics module so it is relatively easy to get the information related to bugs. Finally, we should come up with the information below:

i. The number of bugs in each module for each previous version
ii. The criticality level of the submitted bug

4.3. Merging Change and Defects

Assuming that, we have the analysis of change input and bugs distribution data; if the change information is collected by classifying changes than the following function must be build:

$$N_{bugsexpected} = f\ (A_{features},\ A_{bugfix},\ A_{change},\ A_{size},\ A_{bugsfound},\ N_{features},\ N_{bugfix},\ N_{change},\ N_{size}) \quad (1)$$

Where:

$N_{bugs\ expected}$ is the number of total bugs expected in the next stable version

$A_{features}$ is the array of the total number of features introduced for all past stable versions with the order of version number.

A_{bugfix} is the array of the total number of bug fixes completed for all past stable versions before publishing the version with the order of version number.

A_{change} is the array of the total number of changes completed for all past stable versions with the order of version number.

A_{size}, is the array of size difference of a version with the previous version in terms of kilobytes for all past stable versions with the order of version number.

$A_{bugsfound}$ is the array of the total number of bugs reported for all past stable versions with the order of version number.

$N_{features}$ is the total number of features introduced for the next stable version.

N_{bugfix} is the total number of bug fixes completed for the next stable version.

N_{change} is the total number of changes completed for the next stable version.

N_{size}, is the size difference of the next version with the previous version in terms of kilobytes.

F is a function of these parameters builds up in a way to predict $N_{bugsexpected}$

If the change collection method is based on CVS logs and line changes, then the following formula should be used.

$$N_{bugsexpected} = f\ (A_{linesInserted},\ A_{linesDeleted},\ A_{linesChanged},\ A_{size},\ A_{bugsfound},\ N_{linesInserted},\ N_{linesDeleted},\ N_{linesChanged},\ N_{size}) \quad (2)$$

Where:

$N_{bugs\ expected}$ is the number of total bugs expected in the next stable version

$A_{linesInserted}$ is the array of the total number of lines inserted for all past stable versions with the order of version number.

$A_{linesDeleted}$ is the array of the total number of lines deleted for all past stable versions with the order of version number.

$A_{linesChanged}$ is the array of the total number of line changes completed for all past stable versions with the order of version number.

A_{size}, is the array of size difference of a version with the previous version in terms of kilobytes for all past stable versions with the order of version number.

$A_{bugsfound}$ is the array of the total number of bugs reported for all past stable versions with the order of version number.

$N_{linesInserted}$ is the total number of lines inserted for the next stable version.

$N_{linesDeleted}$ is the total number of lines deleted for the next stable version.

$N_{linesChanged}$ is the total number of line changes completed for the next stable version.

N_{size}, is the size difference of the next version with the previous version in terms of kilobytes.

F is a function of these parameters builds up in a way to predict $N_{bugsexpected}$

4.4. Building the F function

The F function has nine parameters totally. The first five parameters, namely $A_{linesInserted}$, $A_{linesDeleted}$, $A_{linesChanged}$, A_{size}, $A_{bugsfound}$ parameters are used to build up the historical data and figuring out the past experiences whereas the last four parameters, namely $N_{linesInserted}$, $N_{linesDeleted}$, $N_{linesChanged}$, $N_{size,\ are\ used}$ supplying only version specific data in order to be able to use historical experience.

Actually, this function can be divided into two parts. The first part has five parameters that should build up the environment and the second part has last four parameters that can supply input to this environment for predicting the number of bugs.

An alternative approach for function could be interpolation or extrapolation of arrays on polynomials. However, we will work on large datasets so that interpolation or extrapolation algorithms are not the best alternatives for the path to converge.

We therefore have chosen another common approach, neural networks. The first five arrays should be used to build up the neural network and train the network and then the last four parameters should be used to acquire the prediction data from the neural network. The F function will convert into a procedure which has the following steps.

i. Build a feed-forward neural network with the five arrays, where the first four is the input and the last array is the output array.
ii. Train the neural network
iii. Simulate the network over the last four parameters regarding the upcoming version
iv. Return the output of neural network

4.5. Proposed Defect Prediction Framework

We have described completing various tasks in order to build up a successful model. Each task in the proposed model must be correctly accomplished in a sequential manner. We propose a framework to modularize the model and describing the flow.

As we stated in Figure 4.1, the framework consists of four layers. In the first layer, necessary data is collected. The required data is CVS level change, change-feature introduction data and previous versions' defect density information. After collecting these data, it is reorganized and converted to input format. Also, data is consolidated to sum up defect numbers and changes for each module. In the third layer, the normalized data is used to train the selected predictor. Depending on the performance of previous versions' defect predictions, necessary optimizations are done in this layer. Finally, we reach to a state where we can successfully predict.

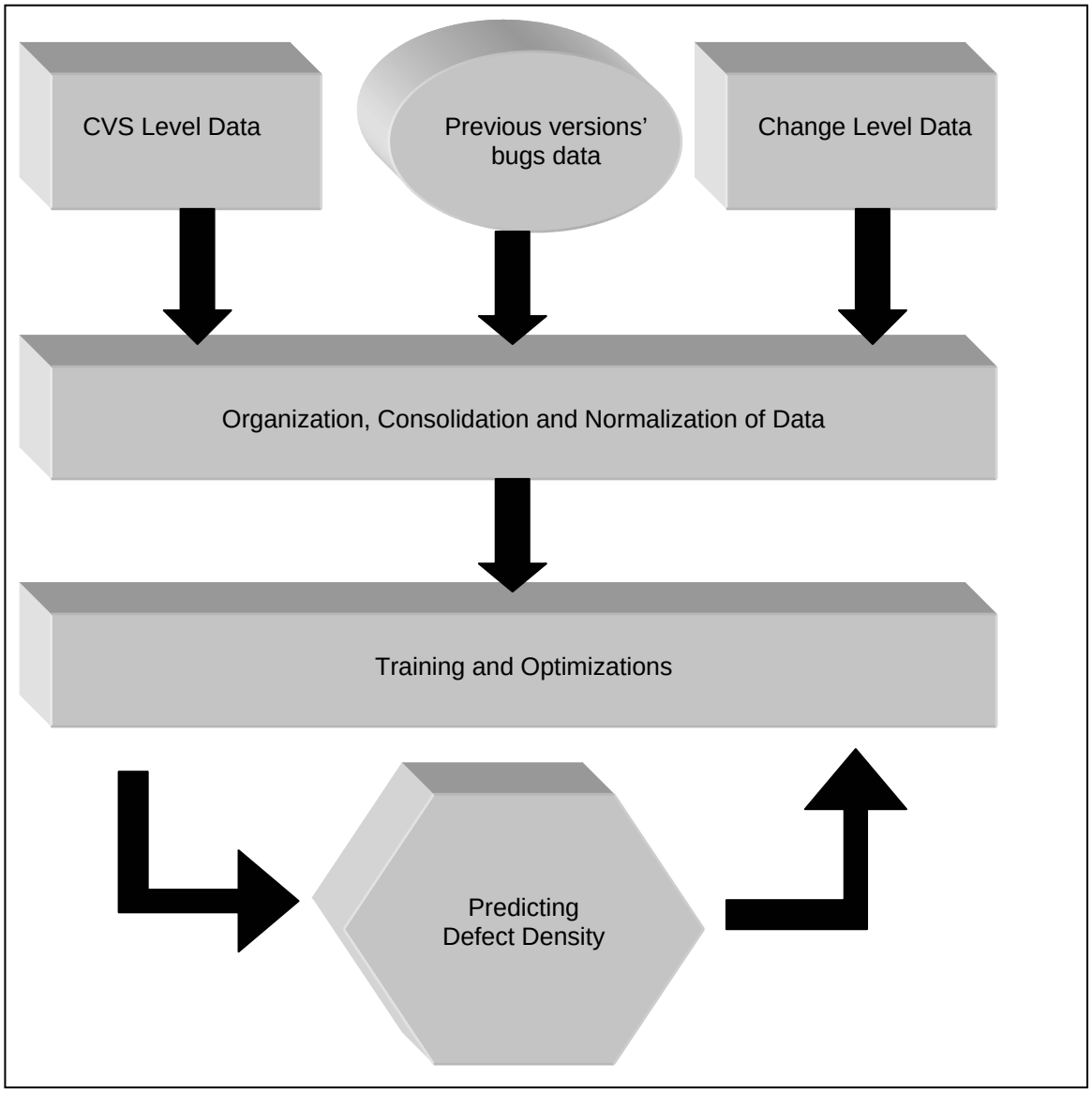


Figure 4.1 Proposed Defect Prediction Framework

Our proposed framework also shows a flowchart for different tasks. In our research, first layer data collection task is done by manually collecting the required data from CVS. But it is possible to integrate an automatic CVS metric collector to the flow.

5. DATA COLLECTION PROCESS

In order to build up a successful model, testing with high quality data is crucial. After proposing the model details, it is important to describe data and its collection procedure to clearly explain the model and general framework.

In this chapter, we will be explaining why we need specific data to experiment the defect prediction solution. In the first section, we will describe the general conventions regarding the data and cumulative actions. This section will also cover what attributes of data are required and what should be the properties of the data.

In the second section 5.2, we will be discussing the data extraction on change analysis aspect. For this purpose an open source project is specially chosen, the motivation behind choosing this specific software will be discussed. Also the metrics about the data source and the reliability of the data will be explained. Also, the objectivity of the data is another important issue in this section.

In the last section, 5.3, we will be discussing the data extraction on CVS logs. The CVS systems keep every single atomic change within the source code of the program. Thus, this dataset will be focusing to a lower layer. We will discuss the advantages and disadvantages of a lower layer as single lines of code.

5.1. Data Requirement Analysis

The core part of this research is the collecting the number of defects in the software. So, collecting the data according to a clear defect definition in the research is important. We have followed this formal definition of defect throughout data collection procedure: “any flaw or imperfection in a software work product or software process”. The defect also called as a flaw or a bug. A defect does not necessarily make a software system fail, it may only degrade its performance or produce an annoying user interface object, where as a defect is also capable of driving a software to total failure [23].

In the light of this definition, the bug tracking listings are the major source for analyzing the bugs, namely defects. Thus any project(s) that will be analyzed in this research must have an extensive bug tracking and classifying tool containing historical data. More importantly this data should be managed in a formal way. If a bug tracking system is considered, anyone can easily enter any kind of bugs without any approval or filter. Such a public system should not be considered as a reliable bug tracking system in the scope of this research.

The selected bug tracking system should also contain at least the following data:

i. The related module or class of the bug
ii. The version of the software that the bug is observed
iii. The classification of the bug (user interface, function interface etc.)
iv. The severity of the bug

The severity of the bug should describe the impact of that specific bug on the software system. A bug might cause a total failure of a software system, whereas another may only degrade the performance.

The related module of the bug should be specified, such that the prediction procedure can be applied at the level of modules. Thus, each module can independently be estimated for defect density.

The severity of the bug is important to calculate the weighted defect density of the software. If the severity level is given weights, it will be more valuable to calculate the weighted defects. Moreover, if the defect density will be used as a metric for stability of a specific version, it is inevitable to distribute the bugs into segments and analyze them in a more meaningful manner.

After all these bug collection, classification and weighted calculation is done on the final dataset. Thus, final dataset should contain the information of every version with the weighted bug densities.

Since our research is focused on the impact of the change among versions, we also require another dataset that contains the change information. Each version change list can be produced in two main approaches. In the first approach, the features, changes and bug fixes are focused at a higher level, where in the second approach only changes in CVS are taken into consideration.

For the first approach, a version's change data should contain these data:

i. Number of new features introduced for each module
ii. Number of changes done for each module
iii. Number of bug fixes completed

"The feature" should be evaluated in the possible smallest piece. The feature can be defined as a "notable property of a software system". Thus, the feature must introduce a better user experience, a nicer way of doing something or must cause an increase in the program requirements definition. The feature for our purpose will be the smallest piece of code that can satisfy the above feature definition. We want to clearly define the feature, thus, we try to prevent calling numerous feature set as a single number of feature. An incorrectly calculated feature number would cause misleading results in the defect prediction process.

Because of the criticality of the definition of feature, it is recommended to complete the identification of features by only an objective team or trained classifiers. Preventing any developer to name a feature should increase the accuracy of the prediction system. The dataset we would use must contain clearly classified feature data and it should be approved by authorized project managers.

The second item in the list is the number of changes done for each module. The difference between change and feature is clear. A "change" is not necessarily notably increases the user experience or does not bring a new functionality, it simply changes the way what a program does. This change could be in user interface, class definitions, or even in algorithms. It is important to mention that notably value-additions in algorithms should be considered as a feature. By keeping the modules that changes or features introduced, our

prediction system will be able to predict not only globally but also at the module level accuracy.

Another important attribute of the data is the number of bug fixes completed. We assumed that every so-called stable version may contain bugs and all the known bugs are fixed for the next stable version. So, the bug fixing information is a crucial data for predicting the stability factor for the forth-coming version.

We are storing bug fix number, so our prediction system would also be able to give substantial information regarding the new bug introduction rate of activities.. Analysis may be helpful to improve the development methodology.

Table 5.1 below is an example of how the collected date should look like:

Table 5.1 Example of Data Format Extracted for Analysis

Version Name	# Bug Fixes	# Features	# Changes	Bugs Found
Version 1	204	59	45	143
Version 2	192	97	88	102

5.2. Data extraction on change analysis

The primary data used in this research come from open source applications. Open source applications are much easier to access their source code and statistical information on their code. Most of the bug tracking systems of open source applications can be viewed as public. So, this bug tracking systems are an important data sources for this research.

Choosing the correct project for data extraction is quite an important issue. After an extensive assessment in open source project, Azureus – Bit torrent client is selected [41].

The general attributes of the project is briefly explained in Figure 5.1.

Figure 5.1 Azureus project details

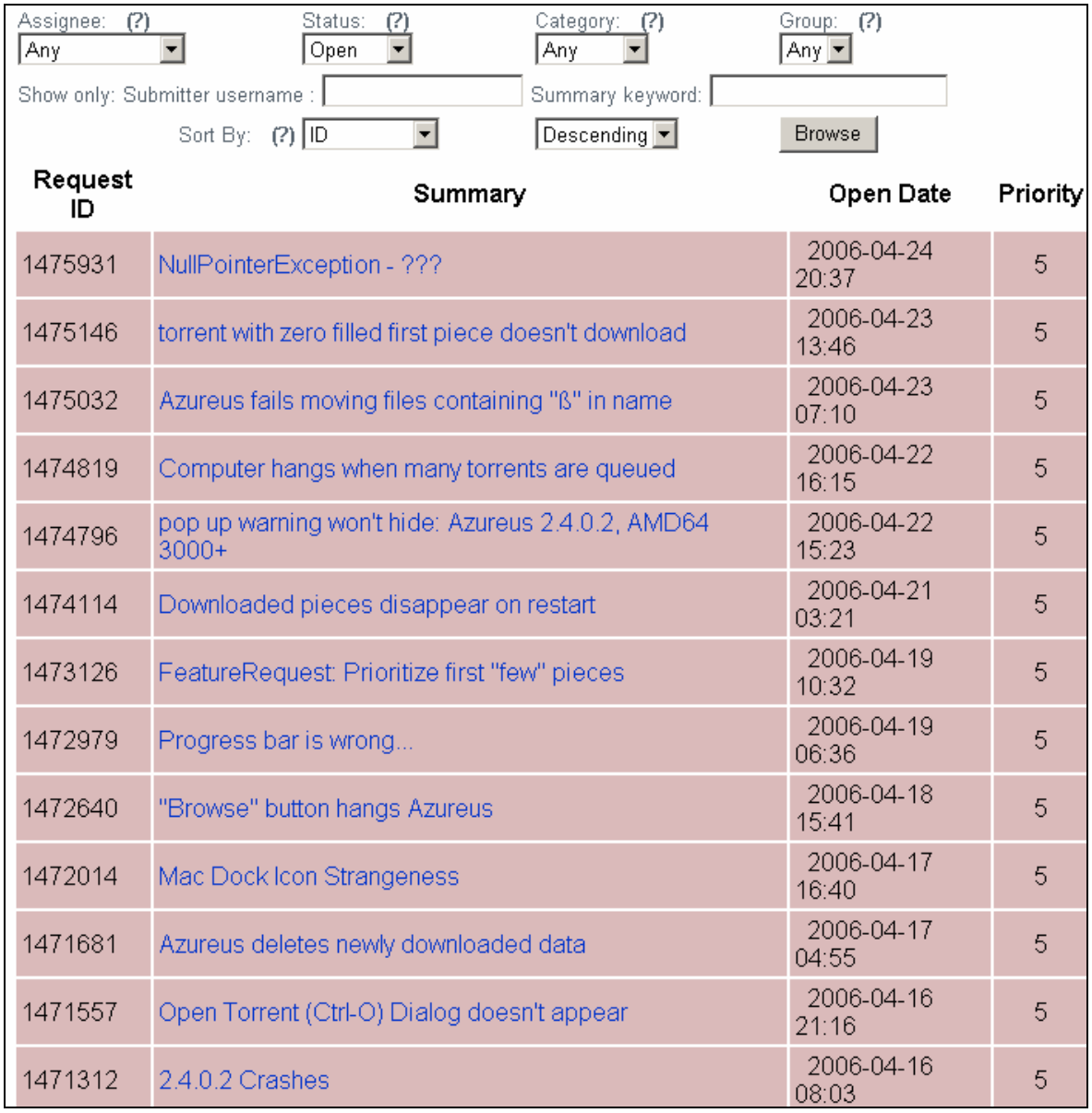
It is important to explain the motivation for selecting Azureus among all alternatives. The most important aspects are:

i. It is the most active project in the open-source community.
ii. There exist 22 developers who actively participate in the development process. Such number of developers is quite high for an open-source project.
iii. It has 28 stable versions to analyze for defects
iv. There are 2322 bugs in total to analyze

v. It is a user-application P2P software. It has wide range of new features.

Because of the reasons listed above, we believe that this project will be a good data source for our proposed defect prediction model.

We have tried to communicate with source forge many times, but they were not kind enough to answer us for data statistics. That is why we collected every single bug detail from their bug tracking software one by one.



Assignee: (?) Any | Status: (?) Open | Category: (?) Any | Group: (?) Any

Show only: Submitter username : | Summary keyword:

Sort By: (?) ID | Descending | Browse

Request ID	Summary	Open Date	Priority
1475931	NullPointerException - ???	2006-04-24 20:37	5
1475146	torrent with zero filled first piece doesn't download	2006-04-23 13:46	5
1475032	Azureus fails moving files containing "ß" in name	2006-04-23 07:10	5
1474819	Computer hangs when many torrents are queued	2006-04-22 16:15	5
1474796	pop up warning won't hide: Azureus 2.4.0.2, AMD64 3000+	2006-04-22 15:23	5
1474114	Downloaded pieces disappear on restart	2006-04-21 03:21	5
1473126	FeatureRequest: Prioritize first "few" pieces	2006-04-19 10:32	5
1472979	Progress bar is wrong...	2006-04-19 06:36	5
1472640	"Browse" button hangs Azureus	2006-04-18 15:41	5
1472014	Mac Dock Icon Strangeness	2006-04-17 16:40	5
1471681	Azureus deletes newly downloaded data	2006-04-17 04:55	5
1471557	Open Torrent (Ctrl-O) Dialog doesn't appear	2006-04-16 21:16	5
1471312	2.4.0.2 Crashes	2006-04-16 08:03	5

Figure 5.2 Interface of bug tracking software in Azureus project

Using the interface shown in Figure 5.2, we have collected all bugs data regarding Azureus project. Then, we have built the list of stable versions and their launch dates as in Table 5.2.

Table 5.2 Stable versions and launch data table

Date	Version
2003-01-01	2.0.0.0
2003-07-28	2.0.1.0
2003-09-03	2.0.2.2
2003-09-07	2.0.2.5
2003-09-08	2.0.2.6
2003-09-12	2.0.2.7
2003-09-30	2.0.3.0
2003-10-12	2.0.3.2
2003-11-11	2.0.4.0
2003-11-19	2.0.4.2
2003-12-18	2.0.6.0
2004-01-31	2.0.7.0
2004-03-07	2.0.8.0
2004-03-14	2.0.8.2
2004-03-15	2.0.8.4
2004-05-29	2.1.0.0
2004-06-20	2.1.0.2
2004-07-07	2.1.0.4
2004-10-31	2.2.0.0
2004-12-18	2.2.0.2
2005-05-02	2.3.0.0
2005-05-25	2.3.0.2
2005-06-27	2.3.0.4
2005-11-22	2.3.0.6
2006-02-10	2.4.0.0

After creating the table, we have mapped all bugs with the stable version. Thus, we acquired a distribution of bugs over the stable versions, as in Figure 5.3.

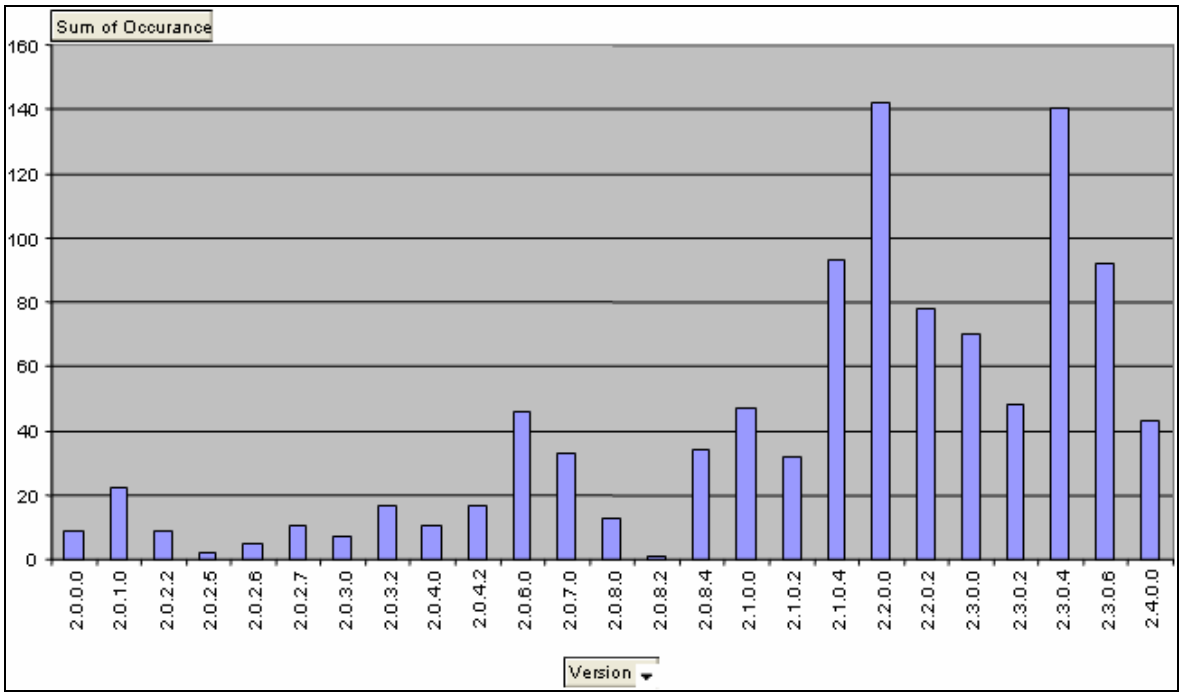


Figure 5.3 Versions vs. bug occurrence

It should be noted that an important property of the bug data is the severity level. So if we segment the bugs into the different severities, the resultant distribution will as in Figure 5.4.

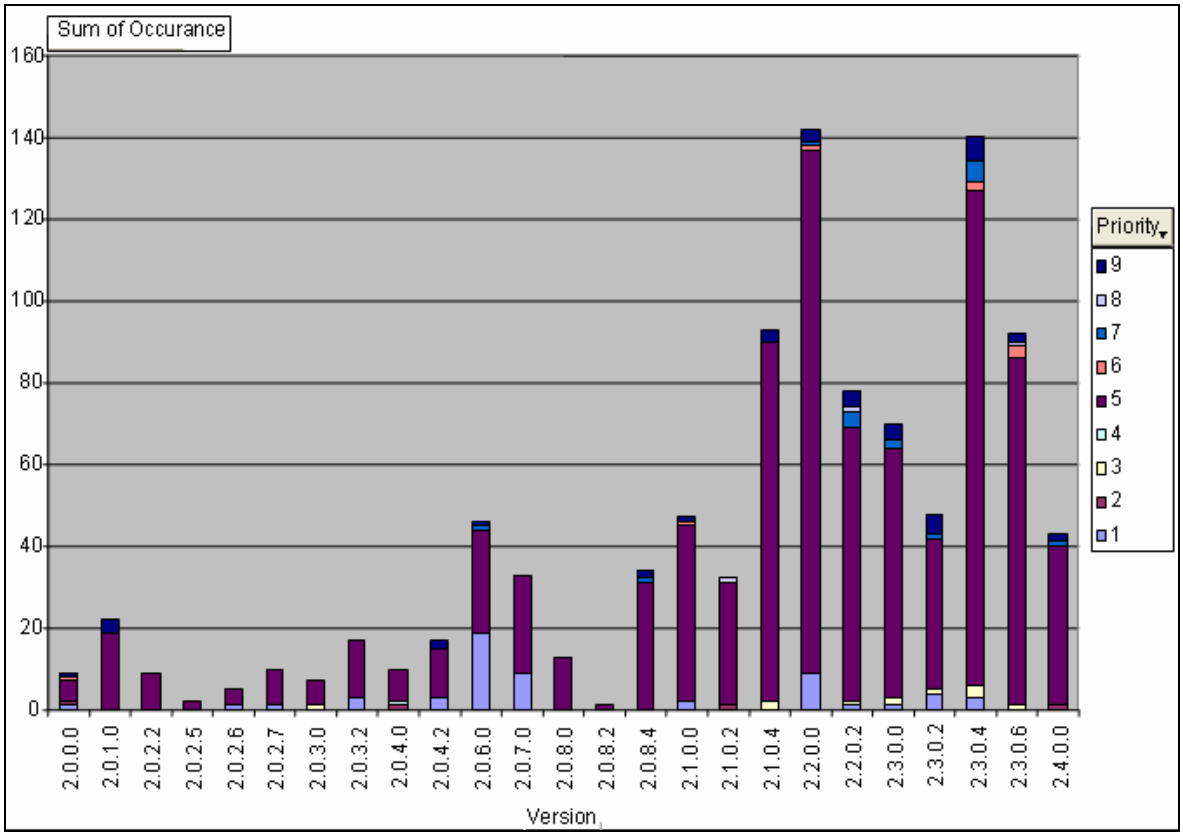


Figure 5.4 Versions vs. sum of occurrence with bugs' priorities

Finally, the weighted distribution is calculated by taking these severity classes into consideration. With this action, we aim to quantify the defect density of the modules in the versions. The last distribution we achieved is in Figure 5.5.

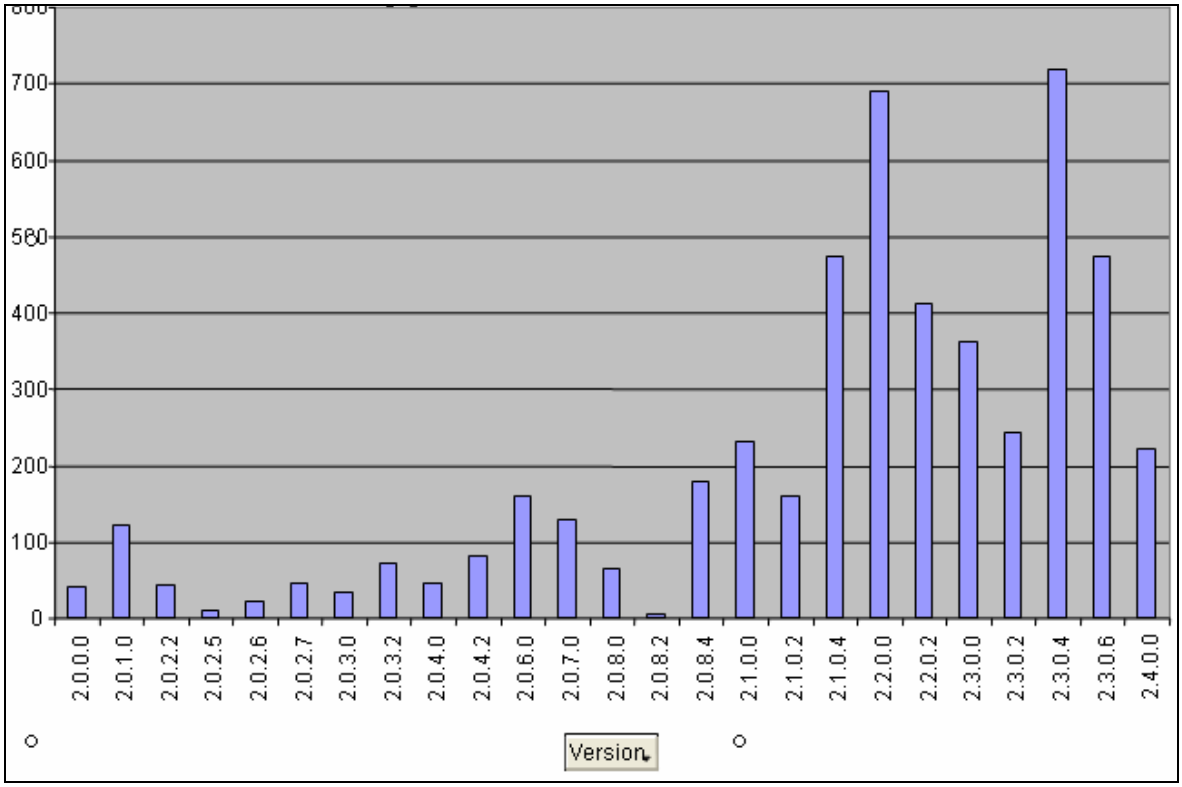


Figure 5.5 Weighted defect densities vs. versions

This dataset and distribution will be the bases for setting the defect density. Each version has its own defect impact rate which also implies the criticalities of the defects.

Another dataset regarding this project required for defect prediction purposes is the change information. Each version has major differences from the previous one in terms of change, bug fix and feature. The CVS logs of this project are scanned in detail and we are able to put through a dataset and distribution of change over the versions. An example subset of collected data is shown in Table 5.3.

Table 5.3 Example dataset of change over versions

Version	Change Type	Module	Differences	Explanation
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Logging separated into sections [TuxPaper]
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Plugins can be disabled from starting up [TuxPaper]
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Separate high-speed transfer rates between peers within the local LAN [Parg,Nolar]
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Encrypted peer connections [Parg,Nolar]
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Revamped and much improved piece-picking code [MjrTom]
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Option to bind outgoing connections to the same local port, may help with NAT router instability [Nolar]
2.4.0.0	FEATURE	Plug	1	FEATURE: Plug | HTTP webseed support (http://www.getright.com/seedtorrent.html) [Parg]
2.4.0.0	FEATURE	Plug	1	FEATURE: Plug | New "team seeder" plugin [Parg]
2.4.0.0	FEATURE	Dev	1	FEATURE: Dev | Plugins can now add views/tabs to Torrent Details, Peers View, etc [TuxPaper]
2.4.0.0	FEATURE	UI	1	FEATURE: UI | Draggable column reordering and column indicator (w/SWT 3.2+) [TuxPaper]
2.4.0.0	FEATURE	UI	1	FEATURE: UI | Peer piece map in Peers Tab [TuxPaper]
2.4.0.0	FEATURE	UI	1	FEATURE: UI | Manual tracker scrape option if auto-scrape disabled [Parg]
2.4.0.0	FEATURE	UI	1	FEATURE: UI | Share-ratio indicator and options to hide the various indicators [Parg]
2.4.0.0	FEATURE	UI	1	FEATURE: UI | Separate per-torrent options panel [Parg]
2.4.0.0	CHANGE	Core	1	CHANGE: Core | Clearer firewalled/NAT status reporting [Nolar]
2.4.0.0	CHANGE	Core	1	CHANGE: Core | Do not open the wiki NAT problem page if firewall status is OK [Nolar]
2.4.0.0	CHANGE	Core	1	CHANGE: Core | Disk manager threads - limited pool now serves all disk read/write requests [Parg]
2.4.0.0	CHANGE	Core	1	CHANGE: Core | Single thread now serves torrent piece picking etc (was one per download) [Parg]
2.2.0.0	FEATURE	UI	1	FEATURE: UI | URL Downloader window now support to set-up referrer and saves last used referrers [Parg, Gudy]
2.2.0.0	FEATURE	UI	1	FEATURE: UI | New Statistic page about the disk cache (yeah more CPU consuming graphs) [Gudy]
2.2.0.0	FEATURE	UI	1	FEATURE: UI | Down/Up speed indicators in main view are now double-clickable to open the Stats View [Gudy]
2.2.0.0	FEATURE	UI	1	FEATURE: UI | Added Path and # Remaining Pieces to Files view, Tracker Name to MyTorrents view [Nolar]
2.2.0.0	FEATURE	Plug	1	FEATURE: Plug | webui support for torrent encoding choice [Parg]
2.2.0.0	FEATURE	Plug	1	FEATURE: Plug | Tracker torrent stats available via xml/http interface [Parg]

For the whole version list a total of 758 lines of data are collected. This data contains module-based change type with an explanation of completed task. Differences column is added as a parameter to be able to give weights to change actual. This data is collected from the CVS and file change log of the Sourceforge.net [41].

The distribution of the data is shown in the Figure 5.6

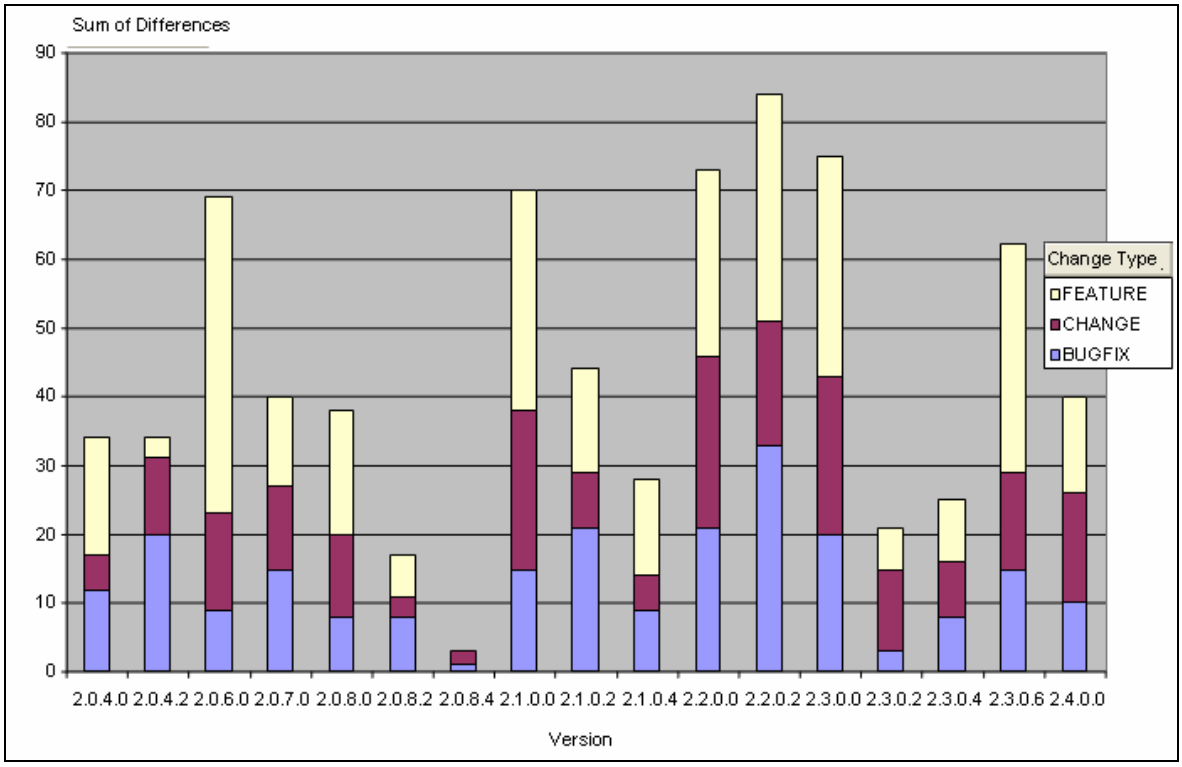


Figure 5.6 Change type distributions vs. versions

The final adaptation of data was merging the change information with the defect information regarding to versions. So, two datasets are merged and grouped into versions. The resultant output is shown in Table 5.4

Table 5.4: Final dataset containing both change and defect information

Version	BugFix	Change	Feature	Size Change	Bugs Introduced
2.0.4.0	12	5	17	319	46
2.0.4.2	20	11	3	15	81
2.0.6.0	9	14	46	495	160
2.0.7.0	15	12	13	184	129
2.0.8.0	8	12	18	268	65
2.0.8.2	8	3	6	29	5
2.0.8.4	1	2	0	102	180
2.1.0.0	15	23	32	609	232
2.1.0.2	21	8	15	623	160
2.1.0.4	9	5	14	122	473
2.2.0.0	21	25	27	482	689
2.2.0.2	33	18	33	312	411
2.3.0.0	20	23	32	1215	362
2.3.0.2	3	12	6	127	244
2.3.0.4	8	8	9	136	718
2.3.0.6	15	14	33	358	472
2.4.0.0	10	16	14	537	222

After this extensive work of data collection this project is ready with change and defect data over versions. The dataset now can be directly used in our proposed model for defect prediction.

5.3. Data extraction on CVS logs

The previous project was a user application for P2P network connections. But in order to test the proposed method for different kind of projects, it is important to collect data from a completely different project type. That is why we have chosen Linux kernels. This project is more delicate in terms of defects and it has a lower defect tolerance limit since Linux is an operating system.

Linux kernels are also open source projects and all the kernel source and CVS system can be found at www.kernel.org [42]. The dataset is chosen among sub releases of 2.4 and 2.6 versions of Linux kernels. Since the even numbered kernels are considered as stable, we also take stability of the versions into consideration.

We also chose the versions older than 6 months, because the defect reporting pool might not be full enough with all defects for the latest versions of kernels. In the end, we come up with 33 versions of Linux kernels to observe in our experiment.

At this point, we wanted to look at CVS level changes instead of feature, change and bug fix analysis. The CVS file difference analysis tool helps us to give the number of changed lines, number of deleted lines and inserted lines. Thus, we have extracted all 33 versions' change information as it is seen in Figure 5.7.

/pub/linux/kernel/v2.6/patch-2.6.16.11.bz2

Show entire file

Makefile	1 +	1 -	0 !
arch/alpha/kernel/setup.c	17 +	0 -	0 !
arch/alpha/kernel/smp.c	3 +	5 -	0 !
arch/i386/kernel/apm.c	1 +	1 -	0 !
arch/i386/kernel/cpu/amd.c	2 +	0 -	0 !
arch/i386/kernel/cpu/cpufreq/Kconfig	1 +	0 -	0 !
arch/i386/kernel/cpu/cpufreq/p4-clockmod.c	1 +	1 -	0 !
arch/i386/kernel/cpu/cpufreq/speedstep-smi.c	3 +	1 -	0 !
arch/i386/kernel/dmi_scan.c	1 +	1 -	0 !
arch/m32r/kernel/m32r_ksyms.c	0 +	4 -	0 !
arch/m32r/kernel/setup.c	5 +	7 -	0 !
arch/m32r/kernel/smpboot.c	10 +	9 -	0 !
arch/m32r/lib/Makefile	2 +	2 -	0 !
arch/m32r/lib/getuser.S	0 +	88 -	0 !
arch/m32r/lib/putuser.S	0 +	84 -	0 !
arch/powerpc/kernel/pci_64.c	1 +	0 -	0 !
arch/powerpc/kernel/setup_64.c	4 +	6 -	0 !
arch/powerpc/kernel/signal_64.c	1 +	1 -	0 !
arch/x86_64/kernel/entry.S	10 +	18 -	0 !
arch/x86_64/kernel/process.c	6 +	2 -	0 !
arch/x86_64/kernel/setup.c	4 +	0 -	0 !
drivers/base/cpu.c	1 +	1 -	0 !
drivers/base/firmware_class.c	4 +	2 -	0 !

Figure 5.7 The CVS log analysis of kernel files between versions

First column shows the file name which is given as input for comparison. The second column shows the number of insertions where as the third column shows the number of lines deleted. At the bottom line, the total number of changes is also displayed. After extensive human work for data extraction, the change data was fully collected.

Another challenge for Linux kernels was extracting the defect information for each version. At this point, we have used the query mechanisms of Linux Kernel Tracker [43]. This site contains every single bug entered for any of the Linux kernel versions. It has extensive statistics and querying options. All the bug data is collected and classified using the panel shown in Figure 5.8.

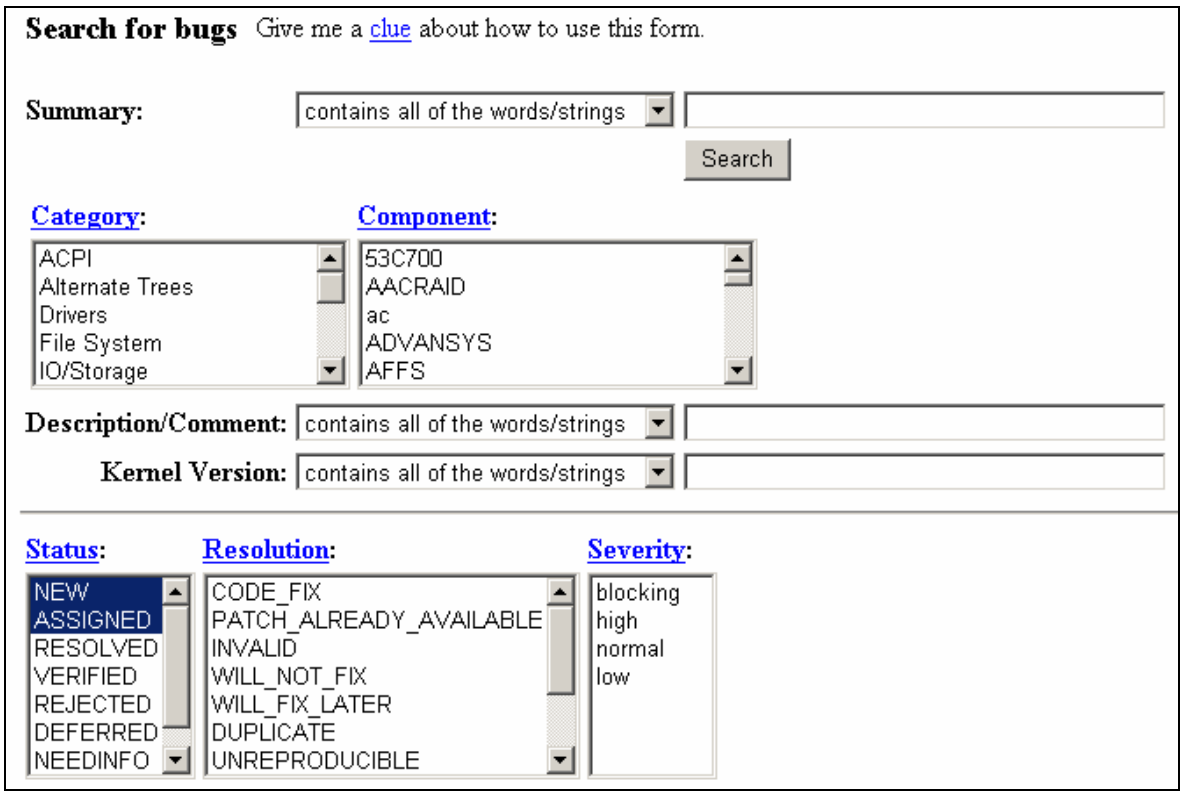


Figure 5.8 Linux bug searching panel

The last data needed to collect was the size change information of Linux kernels. This data was collected using the source code overall size and calculating the differences from version to version. Finally we end up a table for Linux kernel as similar to Figure 5.7.

6. EXPERIMENTS

6.1. Using Neural Networks

In software defect predictions, networks of non-linear elements, interconnected through adjustable weights hold an important place. These networks are called neural networks. Neural networks pretend like biological networks of neurons using non-linear elements have as their inputs a weighted sum of the outputs of other elements [54].

We have chosen to use neural networks for our experiments. Because, neurons have the ability to adapt to a particular situation by changing their weights and thresholds. A feed-forward allows signals to travel one way only; from input to output. [54]. Throughout the experiments in this research, multi layer perceptron (MLP) method is used in neural network experiments. Multilayer perceptrons are feed-forward neural networks trained with the standard back propagation algorithm [54]. The back propagation networks were trained using scaled conjugate gradient optimization in our experiments.

6.2. Deciding Hidden Layers and Units

Bishop[46] and Sonntag[47] claim that, in MLPs with step/threshold/Heaviside activation functions, you need two hidden layers for full generality. On the other hand, if you have only one input, there seems to be no advantage to using more than one hidden layer. According to Sarle, [48]:

> "Unfortunately, using two hidden layers exacerbates the problem of local minima, and it is important to use lots of random initializations or other methods for global optimization. Local minima with two hidden layers can have extreme spikes or blades even when the number of weights is much smaller than the number of training cases. One of the few advantages of standard backprop is that it is so slow that spikes and blades will not become very sharp for practical training times."

After this explanation and considering Bishop and Sonntag [46,47], with the fact that total number of inputs is four, we have decided to use one hidden layer.

There are various proposed methods for deciding the number of units in a hidden layer. Although there is no "silver-bullet" to calculate the best hidden unit number, there exists some rule of thumbs for general cases. Swingler [49] claims that, "you will never

require more than twice the number of hidden units as you have inputs" in an MLP with one hidden layer. For our case, since our number of input units is four, one or two hidden units seem to be enough for this generalization. Berry and Linoff [50] also supports this claim by mentioning “One rule of thumb is that it should never be more than twice as large as the input layer.”

In the light of these references we have decided to use one single hidden layer with two hidden units inside. Such that, the hidden units will be exactly the half of inputs and exactly two times of the number of outputs.

6.3. Building the Neural Network

Our neural network is built by using:

vi. Four input units, in a single input layer
vii. Two hidden units, in a single hidden layer
viii. One output unit, in a single output layer

Figure 6.1 represents our final network:

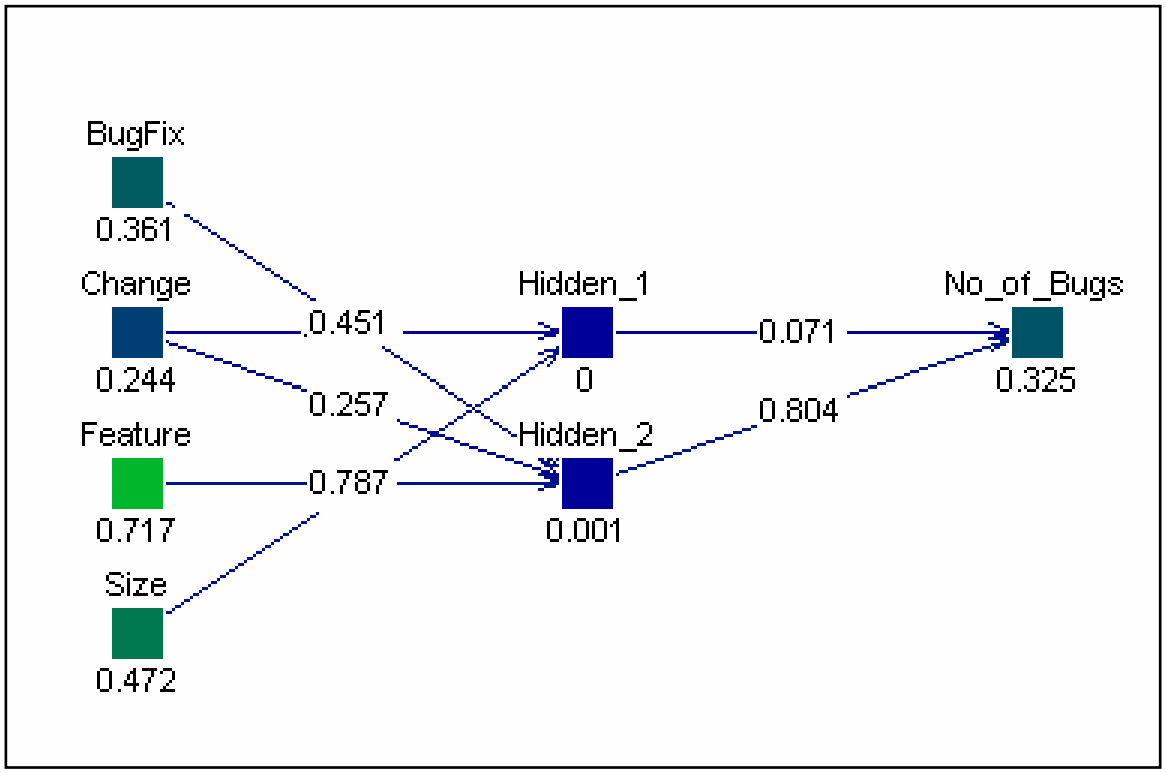


Figure 6.1: Neural network model used for evaluating dataset

In the figure the layer in the left shows the inputs which are bug fix, change, feature and size. The middle layer shows two hidden units. The node at right shows the predicted output as number of bugs. The initial weights are randomly assigned.

6.4. First Data Set

The experiments are accomplished with the first data set which is namely Azureus project. We have accomplished various researches to choose the correct learning algorithm, to decide the learning rate, finding the necessary epoch number and finally the performance results.

6.4.1. Choosing the Correct Learning Algorithm

Various research experiments have been using MLP learning algorithms [1,2,3,7,9]. It is mentioned that MLP and Radial Basis learning functions have better results compared to other learning algorithms. We have decided to use MLP Learning function for training our neural network model.

6.4.2. Deciding Learning Rate

The learning rate parameter is also important to find the correct convergence to the desired output. So, we have tested the learning rate for different inputs. Figure 6.3 shows the epoch versus error analysis for different learning rates.

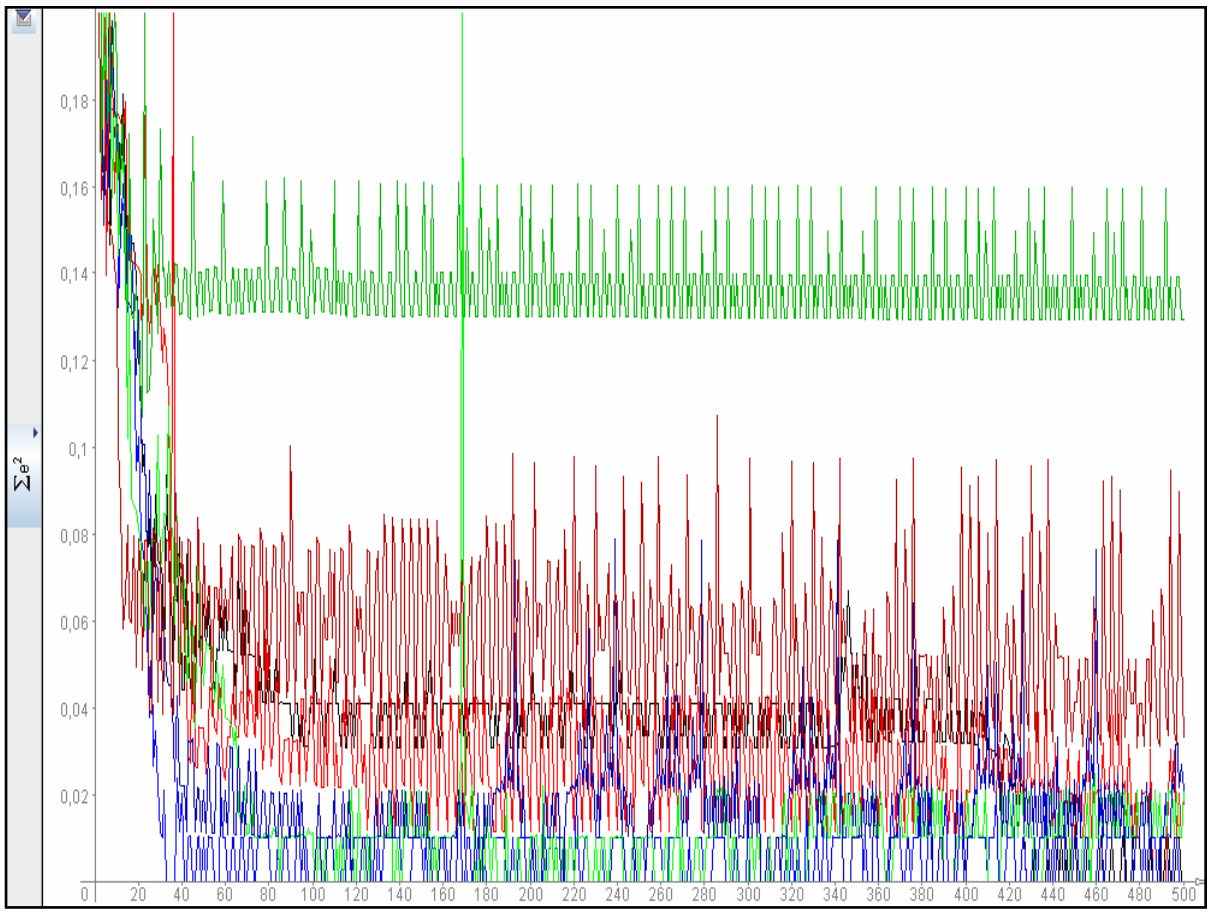


Figure 6.3 Error vs. epoch for different learning rates

The Figure 6.3 can be decoded using the color map below:

i. Black: Learning Rate=0.01
ii. Red: Learning Rate=0.05
iii. Green: Learning Rate=0.10
iv. Blue: Learning Rate n=0.20
v. Dark Blue: Learning Rate=0.30

vi. Maroon: Learning Rate=0.40
vii. Dark Green: Learning Rate=0.50

Analyzing the figure 6.3, we see a plateau of internal error after 40th epoch. Only in the case of 0.5 as learning rate, the internal error cannot decrease below 0.14. Considering the performances, regarding the learning rate 0.2 is selected as best performing with our dataset.

6.4.3. Epoch vs. Error Analysis

After deciding the algorithm and the learning rate, we have analyzed the epoch count and error analysis. Thus, we were able to find minimum required epoch number. Figure 6.4 and 6.5 show this analysis:

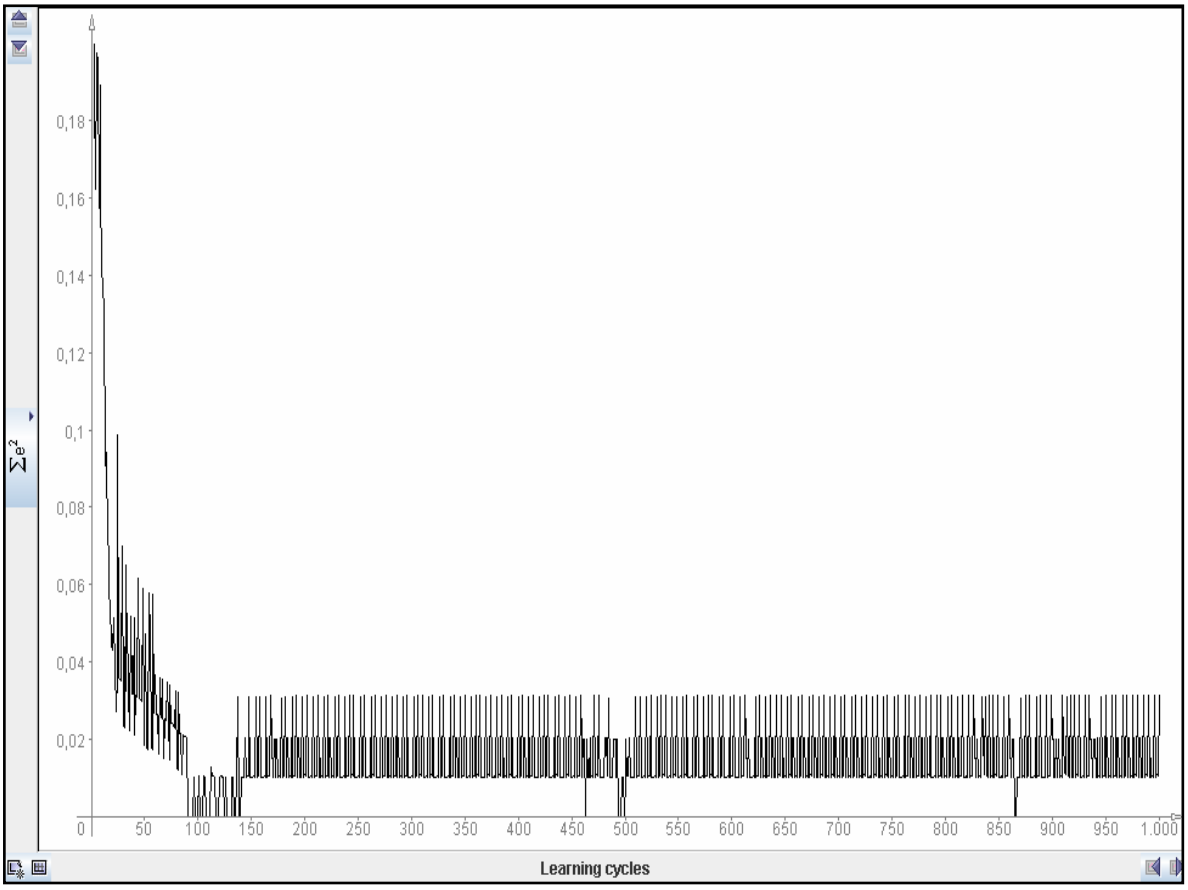


Figure 6.4: Error vs. epoch rate for 1000 epochs

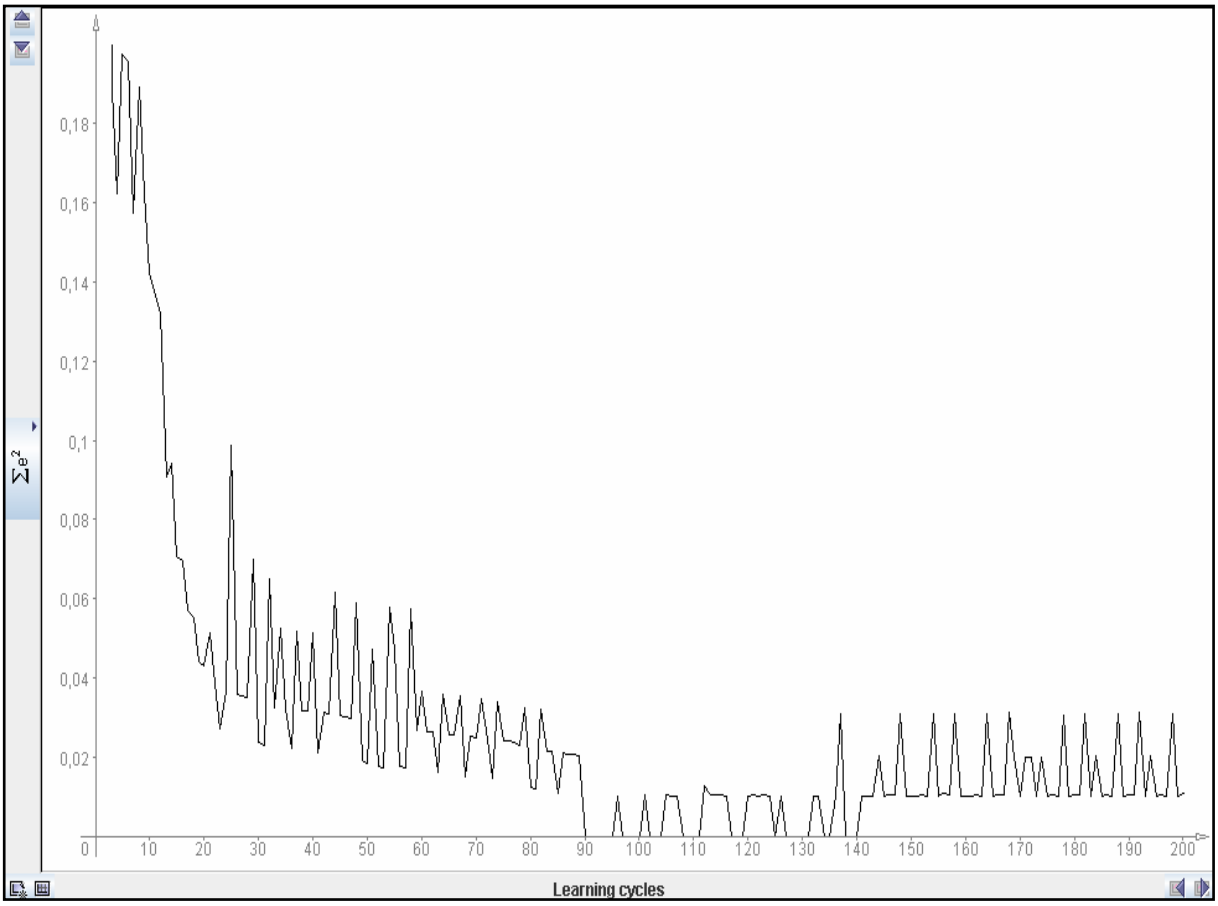


Figure 6.5: Error vs. epoch, detailed

Considering Figure 6.4, we do not observe a decrease in error after 150th epoch. We have zoomed into Figure 6.4 and produced Figure 6.5. We observe that, 90th epoch is the first minimum of error. By putting a safety boundary we have selected the necessary epoch number as 100.

6.4.4. Performance Results

The dataset is used with shuffling, such that all versions were predicted at an instance of multiple runs. The Figure 6.6 shows the error for each version of the product at various executions. Each color represents an execution.

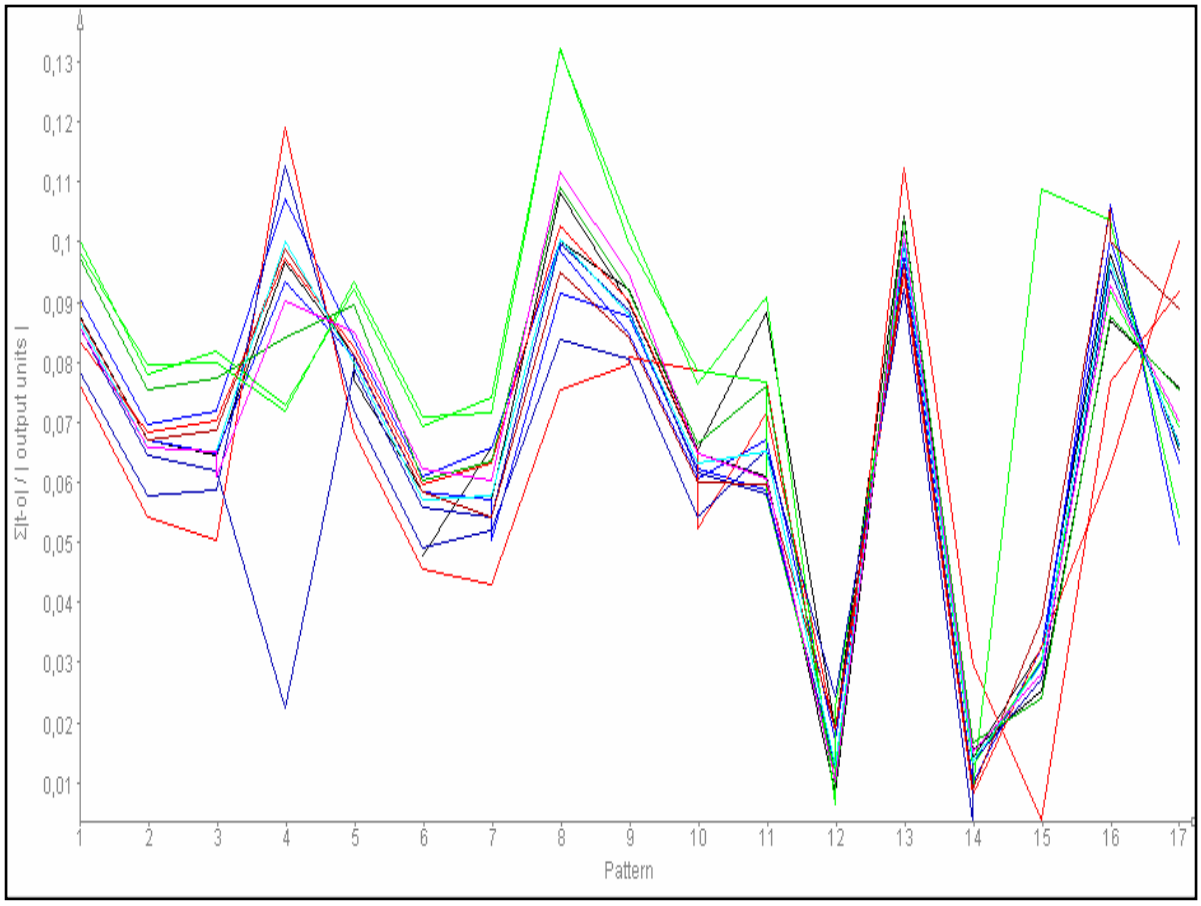


Figure 6.6. Error vs. versions for different executions

We have also completed a regression analysis of predicted and actual data. This analysis can be seen in Figure 6.7. Each execution is represented by a line, we observe that executions carry the same pattern except minor fluctuations. This figure also shows the consistency of executions

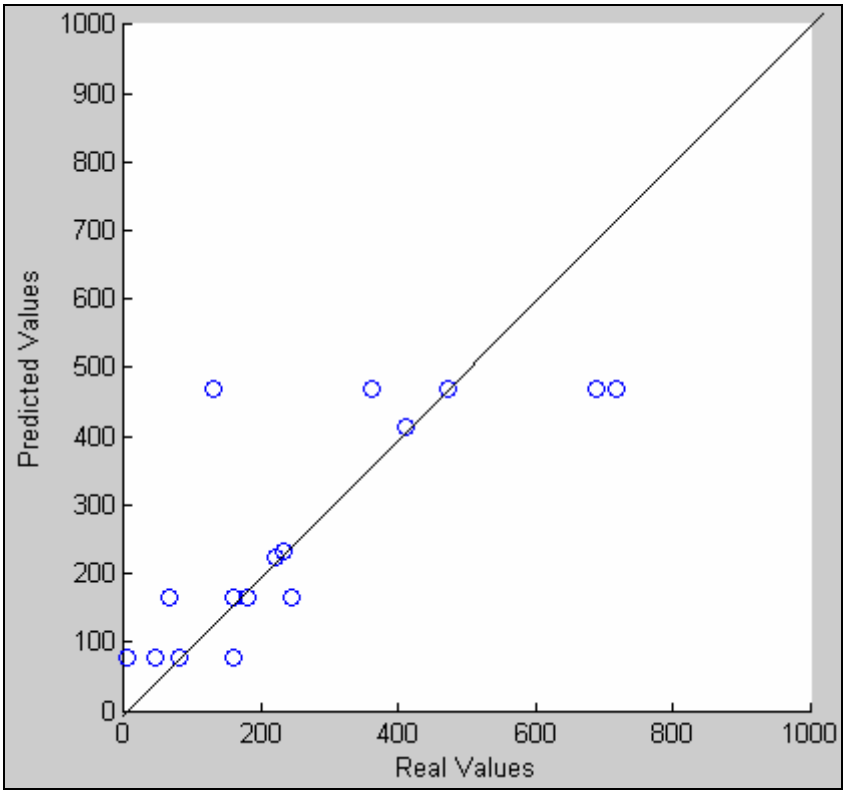


Figure 6.7: Real Values vs. Predicted Values for First Dataset

The execution has been repeated 100 times and the average values are extracted for error. The Table 6.1 contains the real values and the predicted values. The closer points to 45 degree line, represents the more successful prediction. We have small deviations representing the same predicted values.

Table 6.1: Real Values vs. Predicted Values for First Dataset

Real Values	Predicted Values
46	77
81	77
160	166
129	469
65	166
5	77
180	166
232	231
160	77

473	469
689	469
411	412
362	469
244	166
718	469
472	469
222	223

We calculate the Mean Squared Error (MSE) as:

$$MSE = \left(\sum_i \left(Real_i - Predicted_i \right)^2 \right) / N_{test}$$

Where N_{test} = size of test set

Thus,

MSE = 1.7374 e+004

And the variance of test set is:

Var = 4.6323e+004

So, we define a performance metric as:

Performance = MSE / Var

Performance = 0.3751

Since, the performance is linearly correlated with the reciprocal of variance and linearly correlated with MSE, the lower performance metric is the more successful prediction.

6.5. Second Data Set

The second group of experiments conducted on Linux Kernel data. The data collection method and the discussions regarding the data have been completed in Data Collection chapter.

6.5.1. Epoch vs. Error Analysis

We have analyzed the epoch count and error analysis. Thus, we were able to find minimum required epoch number. Figure 6.8 and 6.9 show this analysis:

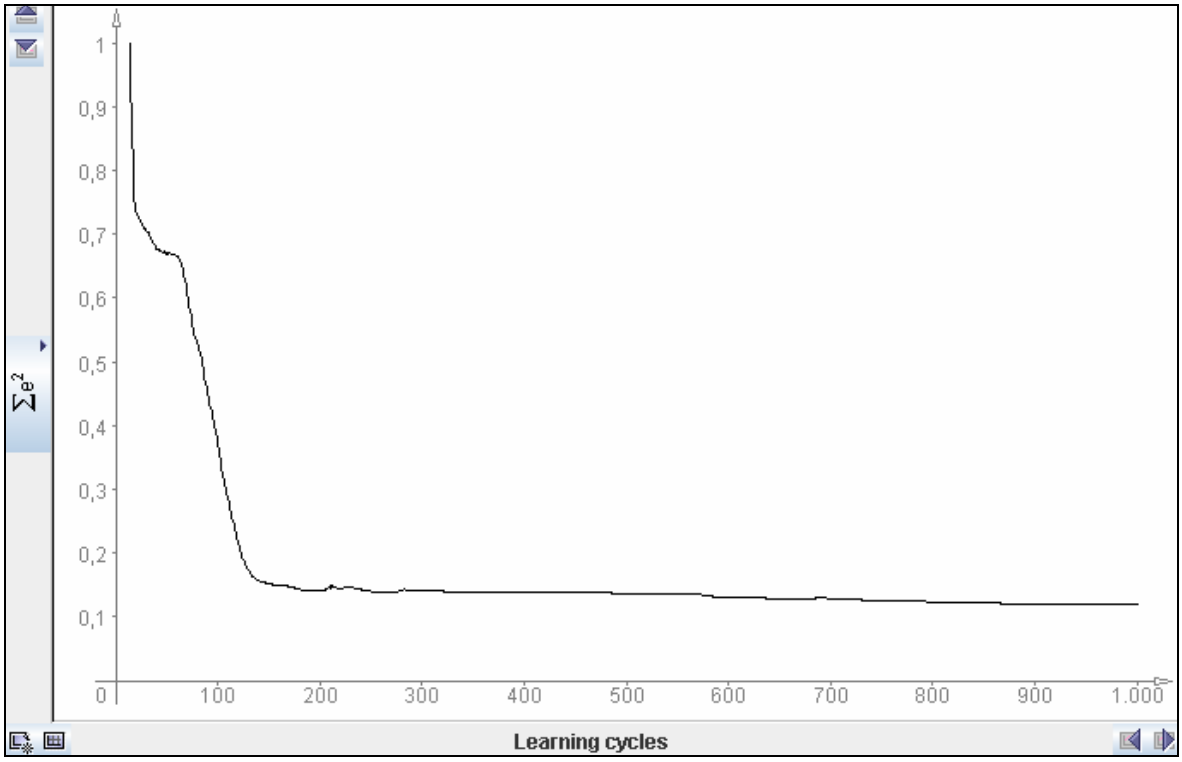


Figure 6.8: Error vs. Epochs for second dataset

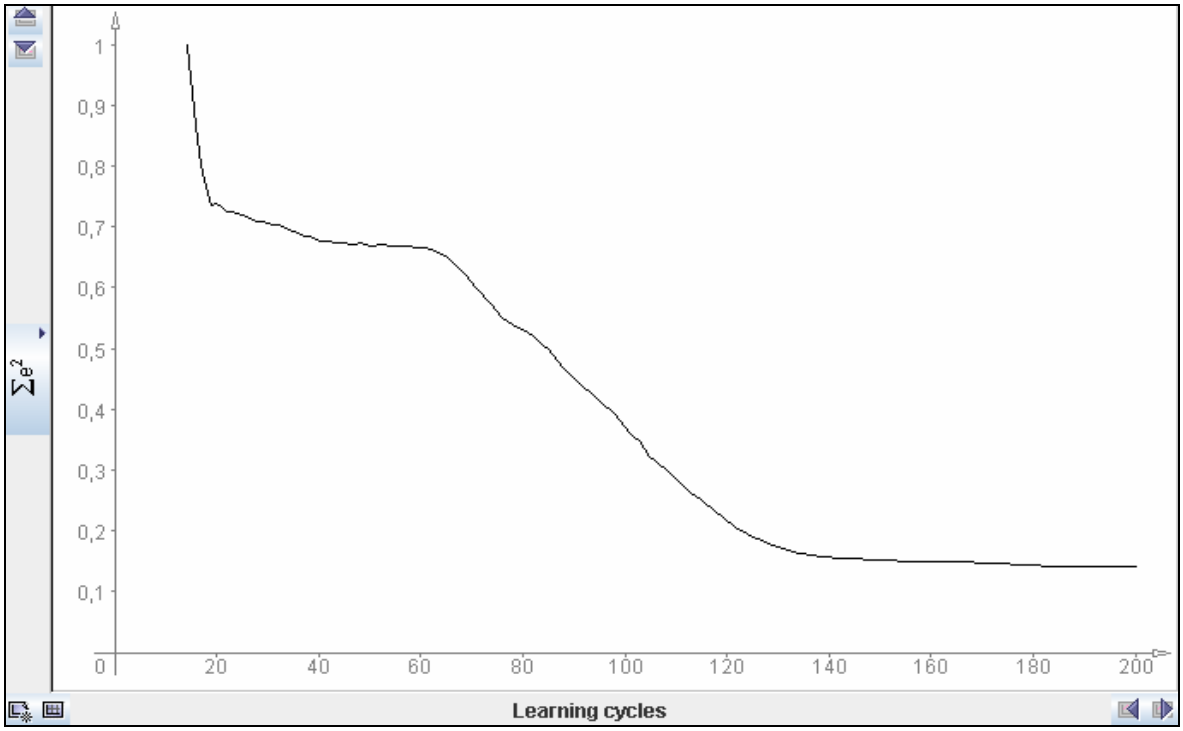


Figure 6.9: Error vs. Epochs for second dataset, detailed

Figure 6.9 is zoomed version of Figure 6.8. Considering the curve follows a plateau after 150 epochs. We have concluded that 200 epochs are required for necessary convergence.

6.5.2. Performance Results

The dataset is used with shuffling, such that all versions were predicted at an instance of multiple runs. The Figure 6.10 shows the error for each version of the product at various executions. Each color represents an execution.

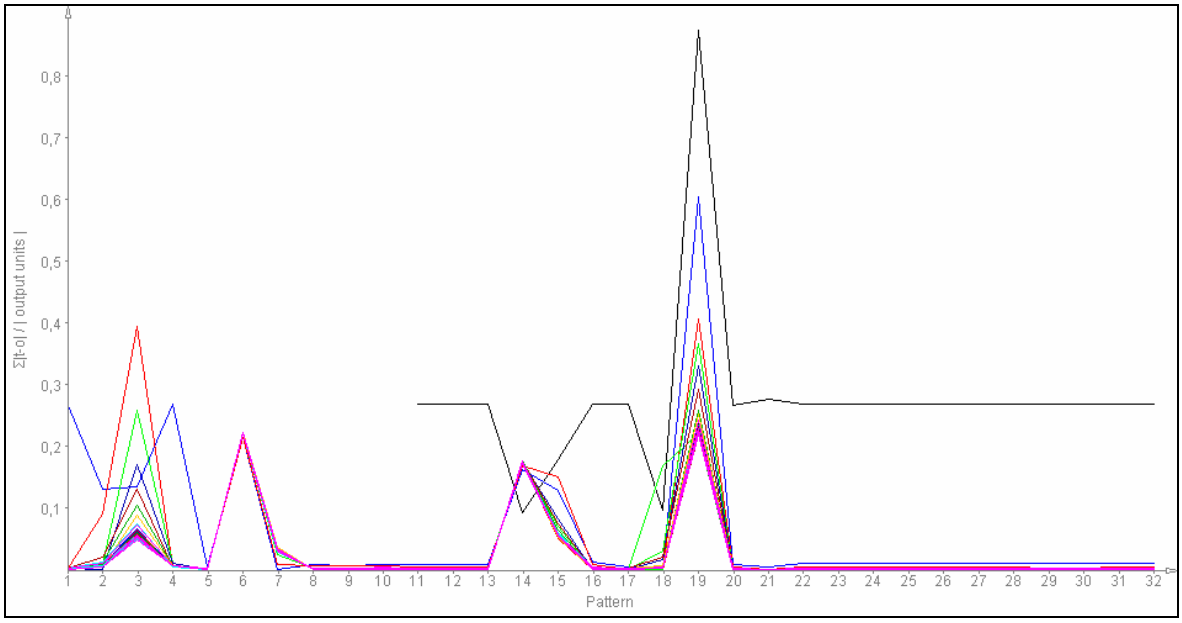


Figure 6.10 Error vs. versions for different executions, second dataset

We have also completed a regression analysis of predicted and actual data. This analysis can be seen in Figures 6.11, 6.12, 6.13.

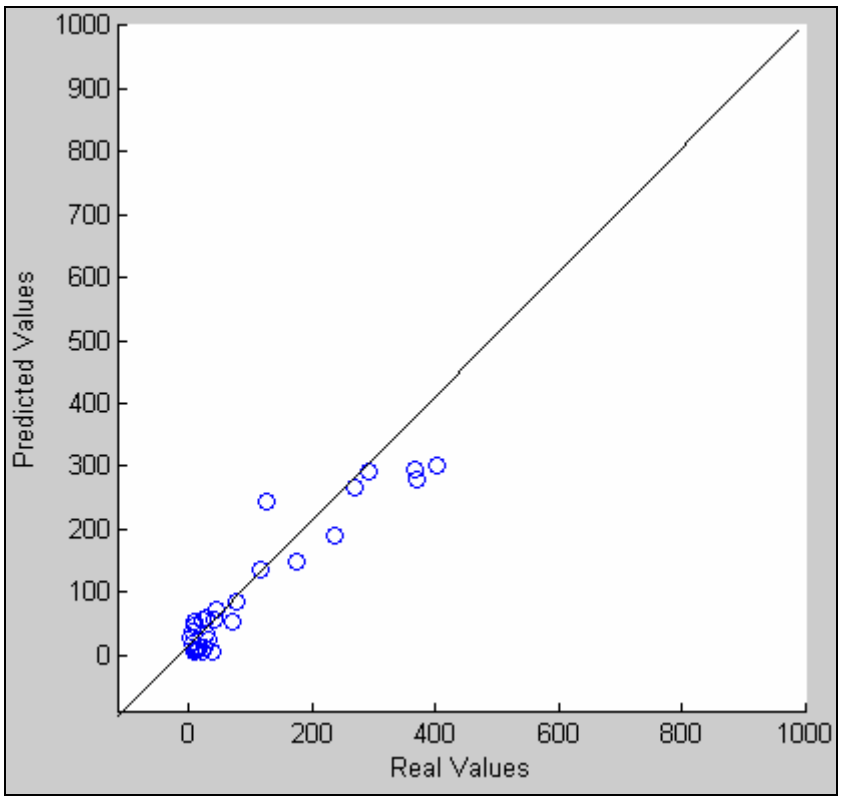


Figure 6.11: Real Values vs. Predicted Values for Second Dataset

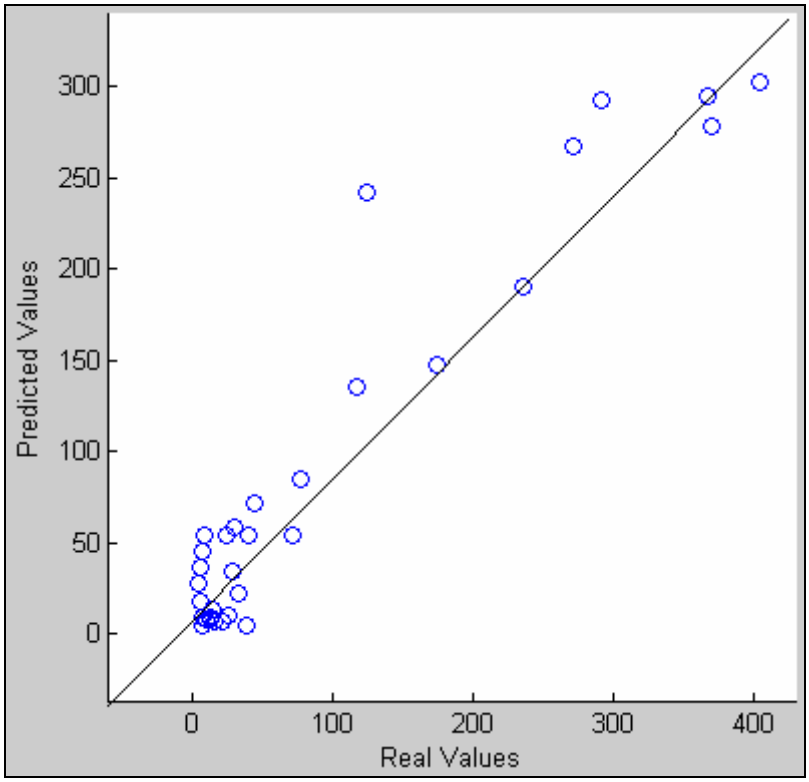


Figure 6.12: Real Values vs. Predicted Values for Second Dataset, detailed

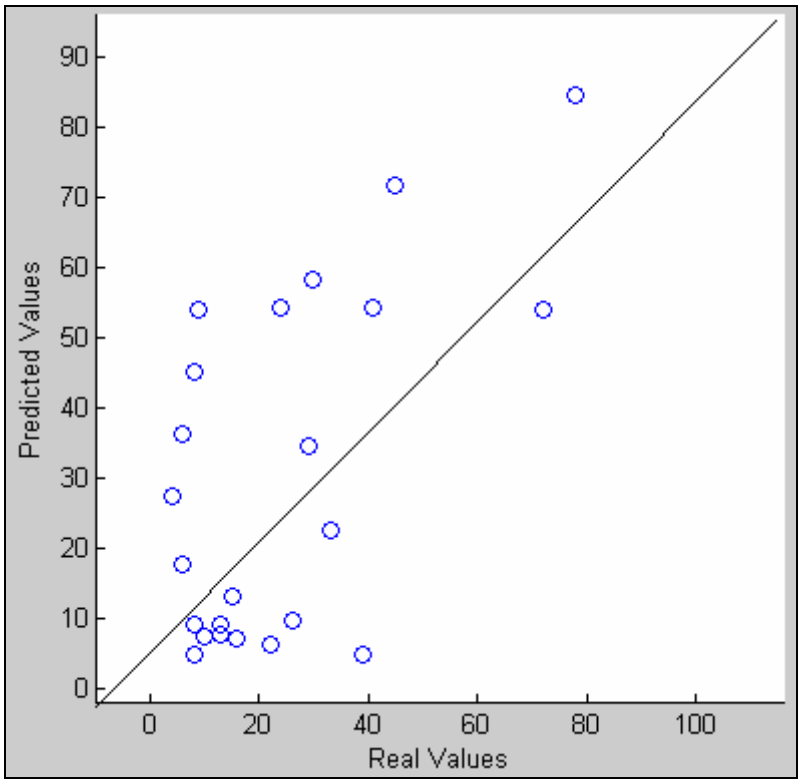


Figure 6.13: Real Values vs. Predicted Values for Second Dataset, Zoom In

The execution has been repeated 100 times and the average values are extracted for error. Table 6.2 contains the real values and the predicted values.

Table 6.2: Real Values vs. Predicted Values for Second Dataset

Real Values	Predicted Values
15	13
9	54
41	54
24	54
72	54
45	72
16	7
30	58
33	23
29	35
6	36
8	45
4	27
8	9
6	18
22	6
78	85
117	135
174	148
125	242
271	267
236	190
291	293
404	302
370	278
367	295
39	5

8	5
10	10
13	13
13	13
26	26

We calculate the Mean Squared Error (MSE) as:

$$MSE = \left(\sum_i (Real_i - Predicted_i)^2 \right) / N_{test}$$

Where N_{test} = size of test set

Thus,

MSE = 1.5880 e+003

And the variance of test set is:

Var = 1.5030 e+004

So, we define a performance metric as:

Performance = MSE / Var

Performance = 0.1057

Note that, the lower performance metric is the more successful prediction.

The research focused on two diverse approaches for defect predictions. The first one uses Azureus project data and collected data has the following dimensions:

i. Number of bug fixes completed
ii. Features introduced
iii. Changes accomplished

iv. Size difference between versions

The dimensions are not directly correlated with code measures, but as we have seen in the experiment they are directly related with the number of bugs. This experiment conducted with large number of bugs for all versions. The average performance metric is calculated as 0.3751.

The second experiment is accomplished with a completely different dataset. This dataset is collected from stable Linux kernel versions, and from kernel bugs databases. This dataset was about two times larger than the previous dataset. The major difference of second experiment is the input dimensions introduced to the model. The input dimensions were:

i. Number of lines inserted
ii. Number of lines deleted
iii. Number of lines changed
iv. Size difference between versions

The major difference of second experiment is, the focus on code change. The second experiment has focused only changes in Code Versioning System level, where as the first experiment considers changes in feature, and bug fix abstract level.

The second experiment's average performance metric was 0.1057, which is more successful than the first experiment.

7. CONCLUSION

In this chapter, we will be stressing the contributions throughout this research. Our contributions are in both data consolidation and model dimensions. Finally, the possible future extension of the model will be analyzed. The model has various possible extension directions to be discussed.

7.1. Contributions

This research contributes to literature on three different aspects. These are data, method and experiment aspects which are clearly explained below.

The first contribution is extensive data on bugs and changes over versions for two different software products. The data collection process was an important milestone throughout this research. The collected data has been archived in a structured way. The data contains important statistical information regarding software version life cycles. Some of the statistical analysis has been mentioned in the data collection part.

The change data has been added into this thesis in Appendix chapters. Since the defects data was huge it was not possible to fit it into Appendix chapter. In addition to that, this data can be used by other researches for their experiments and studies.

The major contribution of this research was to bring an innovative method for defect prediction systems. Introducing the concept of versioning to defect prediction systems, and using difference between versions was an important issue in the scope of this research. Choosing the correct mathematical framework and decision process of algorithms should also be considered as contribution of this research.

Another contribution of this research is two different experiments help us to measure the validity and usability of the model. Conducting two experiments and comparing the performances indicated the possibility of future extensions. Experiments should help quantitatively evaluate the proposed model and compare two sub approaches of the model.

There exist various practical contributions of this research to software development companies. Assuming that, these companies are using a well defined metric collection framework. The proposed model might continuously be applied for upcoming version control. Targeting a defect density level for new version and tracking the density with our proposed model can help software companies to act on a timely basis. Besides, they would have the chance of distributing their testing efforts in a more efficient way. Knowing the modules with higher defect density can prioritize the testing tasks. Optimizing the test phase would result in cost savings.

7.2. Future Work

The method discussed in this research, has been tested with two different approaches. Those were the CVS level approach and, feature level approach. We have experimented these approached independently. An important future work can be using a synthesis of these two approaches and building up a more detailed model which contains both CVS level inputs and feature level inputs. We believe this hybrid model has an important potential to predict defects more successfully.

The research is focused on multi-version software. Thus, we are expecting previous stable versions to start defect predictions. A future work might be extending this model to single version software. Such that, there will not be a prerequisite of previous stable versions to start defect predictions.

8. APPENDIX A: SOURCE CODE

8.1. MLP Driver

```
function outarg = EC_MLP(trainSize, testSize, hiddenUnits, trainingCycles)
addpath('D:\Documents\Akademik\MS\CmpE 690 - Thesis\MatlabLibrary')
%Function Parameters
trainSize=32; %Azureus 17 - Linux 32  - Linux2 21
testSize=0;
hiddenUnits=2;
trainingCycles=500;

classLabelColumn = 5;
featureRange = [1:4];
% Extract data from cvs file
%dataAll = dlmread('Azureus.csv', ';');
dataAll = dlmread('Linux.csv', ';');
%dataAll = dlmread('LinuxSmall.csv', ';');

% Get training class information
classTr = dataAll(1:trainSize,classLabelColumn);
% Get training part of the data
dataTr = mynorm(dataAll(1:trainSize, featureRange));

% Get test class information
classTe = dataAll(trainSize+1:trainSize+testSize,classLabelColumn);


% Get test part of the data
dataTe = mynorm(dataAll(trainSize+1:trainSize+testSize, featureRange));
```

```
%%%
```

```
% function.

%options(9) = 1;

% OPTIONS(10) returns the total number of function evaluations
% (including those in any line searches).

%options(10) = trainingCycles;



%
% OPTIONS(11) returns the total number of gradient evaluations.

%options(11) = trainingCycles;
%
% OPTIONS(14) is the maximum number of iterations; default 100.
%options(14) = trainingCycles;



% 'quasinew', 'scg', 'conjgrad', 'graddesc'
[net, options] = netopt(net, options, dataTr, classTr, 'scg');



y = mlpfwd(net, dataTr);
yt = mlpfwd(net, dataTe);

figure
scatter(classTe,yt); % Test

axis([0 1000 0 1000]);
axis square;
xlabel('Real Values');
ylabel('Predicted Values');
```

```
title('MLP');
mse = sum((classTe - yt).^2) / testSize


classTe
yt
variance = var(classTe)

mse

performance = mse / variance


function x = mynorm(y)
   x = y - (ones(size(y,1),1) * mean(y));
   x = x ./ (ones(size(y,1),1) * (std(y) + 0.00000000000000000000001));
```

8.2. MLP Base Function

```
function net = mlp(nin, nhidden, nout, func, prior, beta)
%MLP Create a 2-layer feedforward network.
%
% Description
% NET = MLP(NIN, NHIDDEN, NOUT, FUNC) takes the number of inputs,
% hidden units and output units for a 2-layer feed-forward network,
% together with a string FUNC which specifies the output unit
% activation function, and returns a data structure NET. The weights
% are drawn from a zero mean, unit variance isotropic Gaussian, with
% varianced scaled by the fan-in of the hidden or output units as
% appropriate. This makes use of the Matlab function RANDN and so the
% seed for the random weight initialization can be  set using
% RANDN('STATE', S) where S is the seed value.  The hidden units use
% the TANH activation function.
%
% The fields in NET are
%   type = 'mlp'
%   nin = number of inputs
%   nhidden = number of hidden units
%   nout = number of outputs
%   nwts = total number of weights and biases
%   actfn = string describing the output unit activation function:
%       'linear'
%       'sigmoid'
%       'softmax'
```

```
%  w1 = first-layer weight matrix
%  b1 = first-layer bias vector
%  w2 = second-layer weight matrix
%  b2 = second-layer bias vector
% Here W1 has dimensions NIN times NHIDDEN, B1 has dimensions 1 times
%NHIDDEN, W2 has dimensions NHIDDEN times NOUT, and B2 has dimensions
%1 times NOUT.
%
%NET = MLP(NIN, NHIDDEN, NOUT, FUNC, PRIOR), in which PRIOR is a
%scalar, allows the field NET.ALPHA in the data structure NET to be
%set, corresponding to a zero-mean isotropic Gaussian prior with
%inverse variance with value PRIOR. Alternatively, PRIOR can consist
%of a data structure with fields ALPHA and INDEX, allowing individual
%Gaussian priors to be set over groups of weights in the network. Here
%ALPHA is a column vector in which each element corresponds to a
%separate group of weights, which need not be mutually exclusive.  The
%membership of the groups is defined by the matrix INDX in which the
%columns correspond to the elements of ALPHA. Each column has one
%element for each weight in the matrix, in the order defined by the
%function MLPPAK, and each element is 1 or 0 according to whether the
%weight is a member of the corresponding group or not. A utility
%function MLPPRIOR is provided to help in setting up the PRIOR data
%structure.
%
%NET = MLP(NIN, NHIDDEN, NOUT, FUNC, PRIOR, BETA) also sets the
%additional field NET.BETA in the data structure NET, where beta
%corresponds to the inverse noise variance.
%
%See also
%MLPPRIOR, MLPPAK, MLPUNPAK, MLPFWD, MLPERR, MLPBKP, MLPGRAD
%

%Copyright (c) Christopher M Bishop, Ian T Nabney (1996, 1997)

net.type = 'mlp';
net.nin = nin;
net.nhidden = nhidden;
net.nout = nout;
net.nwts = (nin + 1)*nhidden + (nhidden + 1)*nout;

actfns = {'linear', 'logistic', 'softmax'};

if sum(strcmp(func, actfns)) == 0
  error('Undefined activation function. Exiting.');
else
  net.actfn = func;
end

if nargin > 4
  if isstruct(prior)
    net.alpha = prior.alpha;
    net.index = prior.index;
  elseif size(prior) == [1 1]
    net.alpha = prior;
  else
    error('prior must be a scalar or a structure');
  end
end

net.w1 = randn(nin, nhidden)/sqrt(nin + 1);
```

```
net.b1 = randn(1, nhidden)/sqrt(nin + 1);
net.w2 = randn(nhidden, nout)/sqrt(nhidden + 1);
net.b2 = randn(1, nout)/sqrt(nhidden + 1);

if nargin == 6
  net.beta = beta;
end
```

Neural Network Model Source

SNNS network definition file V1.4-3D

generated at Sat May 06 14:53:28 2006

network name : Azureus

source files :

no. of units : 7

no. of connections : 10

no. of unit types : 2

no. of site types : 2

learning function : JE_Rprop

update function : Topological_Order

site definition section :

site name	site function
excite	Site_WeightedSum
inhibit	Site_Pi

type definition section :

name	act func	out func	sites

```
-------------|-------------|-------------|---------
LongeroutType | Act_Logistic | Out_Identity |
outType       | Act_Logistic | Out_Identity |
-------------|-------------|-------------|---------
```

unit default section :

```
act      | bias     | st | subnet | layer | act func     | out func
---------|----------|----|--------|-------|--------------|-------------
 0.00000 |  0.00000 | h  |      0 |     1 | Act_Logistic | Out_Identity
---------|----------|----|--------|-------|--------------|-------------
```

unit definition section :

```
no. | typeName       | unitName    | act       | bias      | st | position | act func | out func |
sites
----|----------------|-------------|-----------|-----------|----|----------|----------|----------|---------
  1 |                | BugFix      | 362.00000 |  0.68731 | i  | 1,1,1    |||
  2 |                | Change      | 244.00000 |  0.99359 | i  | 1,2,1    |||
  3 |                | Feature     | 718.00000 |  0.99939 | i  | 1,3,1    |||
  4 |                | Size        | 472.00000 |  0.22300 | i  | 1,4,1    |||
  5 |                | Hidden_1    |  1.00000 |  0.00785 | h  | 4,2,1    |||
  6 |                | Hidden_2    |  1.00000 |  0.00805 | h  | 4,3,1    |||
  7 |                | No_of_Bugs |  1.00000 |  4.80444 | o  | 7,2,1    |||
----|----------------|-------------|-----------|-----------|----|----------|----------|----------|---------
```

connection definition section :

```
target | site    | source:weight
```

-------|---------|--

---

5 | | 4: 0.07066, 3: 0.11777, 2: 0.02358, 1: 0.08707

6 | | 4: 0.07243, 3: 0.12072, 2: 0.02417, 1: 0.08911

7 | | 6: 5.32719, 5: 4.79916

-------|---------|--

---

layer definition section :

layer | unitNo.

------|--

---

2 | 5, 6

3 | 7

------|--

---

9. APPENDIX B: DATA

9.1. Azureus Change Data

Version	Change Type	Module	Diff.	Explanation
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Logging seperated into sections [TuxPaper]
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Plugins can be disabled from starting up [TuxPaper]
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Separate high-speed transfer rates between peers within the local LAN [Parg,Nolar]
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Encrypted peer connections [Parg,Nolar]
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Revamped and much improved piece-picking code [MjrTom]
2.4.0.0	FEATURE	Core	1	FEATURE: Core | Option to bind outgoing connections to the same local port, may help with NAT router instability [Nolar]
2.4.0.0	FEATURE	Plug	1	FEATURE: Plug | HTTP webseed support (http://www.getright.com/seedtorrent.html) [Parg]
2.4.0.0	FEATURE	Plug	1	FEATURE: Plug | New "team seeder" plugin [Parg]
2.4.0.0	FEATURE	Dev	1	FEATURE: Dev | Plugins can now add views/tabs to Torrent

				Details, Peers View, etc [TuxPaper]
2.4.0.0	FEATURE	UI	1	FEATURE: UI	Draggable column reordering and column indicator (w/SWT 3.2+) [TuxPaper]
2.4.0.0	FEATURE	UI	1	FEATURE: UI	Peer piece map in Peers Tab [TuxPaper]
2.4.0.0	FEATURE	UI	1	FEATURE: UI	Manual tracker scrape option if auto-scrape disabled [Parg]
2.4.0.0	FEATURE	UI	1	FEATURE: UI	Share-ratio indicator and options to hide the various indicators [Parg]
2.4.0.0	FEATURE	UI	1	FEATURE: UI	Separate per-torrent options panel [Parg]
2.4.0.0	CHANGE	Core	1	CHANGE: Core	Clearer firewalled/NAT status reporting [Nolar]
2.4.0.0	CHANGE	Core	1	CHANGE: Core	Do not open the wiki NAT problem page if firewall status is OK [Nolar]
2.4.0.0	CHANGE	Core	1	CHANGE: Core	Disk manager threads - limited pool now serves all disk read/write requests [Parg]
2.4.0.0	CHANGE	Core	1	CHANGE: Core	Single thread now serves torrent piece picking etc (was one per download) [Parg]
2.4.0.0	CHANGE	Core	1	CHANGE: Core	Persistent IP bans now have limited lifetime [Parg]
2.4.0.0	CHANGE	Core	1	CHANGE: Core	Reworked compact storage type to support migration of formats [Parg]
2.4.0.0	CHANGE	Core	1	CHANGE: Core	Less requirements to pause downloads when changing file priorities [Parg]
2.4.0.0	CHANGE	Core	1	CHANGE: Core	Only one torrent can now be checking at a time [Parg]
2.4.0.0	CHANGE	Core	1	CHANGE: Core	DHT size estimation improved [Parg]
2.4.0.0	CHANGE	Core	1	CHANGE: Core	Default listen port number now randomized for new installs [Nolar]
2.4.0.0	CHANGE	Core	1	CHANGE: Core	Interrupted "seeding+checking" files now rechecked on startup [Parg]
2.4.0.0	ChANGE	Core	1	ChANGE: Core	Under unix, use all-lowercase ~/.azureus/ as the config dir [Nolar]
2.4.0.0	CHANGE	UI	1	CHANGE: UI	Seeding+checking completeness now shown in status [Parg]
2.4.0.0	CHANGE	UI	1	CHANGE: UI	Faster filling of tables (Mac/some Linux) [TuxPaper]
2.4.0.0	CHANGE	UI	1	CHANGE: UI	Opening torrents changed to use one opener dialog (w/options to change destination filenames, disable files, etc) [TuxPaper]
2.4.0.0	CHANGE	UI	1	CHANGE: UI	Find-As-You-Type in 'My Torrents' changed to Filter-As-You-Type. Ctrl-BS to clear filter. [TuxPaper]
2.4.0.0	BUGFIX	Core	1	BUGFIX: Core	Fix transfer stall bug when MTU size is greater than max upload/download rate [Nolar]
2.4.0.0	BUGFIX	Core	1	BUGFIX: Core	Improved peer connection counting [Parg]
2.4.0.0	BUGFIX	Core	1	BUGFIX: Core	Tracker wasn't handling "accept-encoding" properly [Parg]
2.4.0.0	BUGFIX	Core	1	BUGFIX: Core	Multiple file renames/retargets wasn't working [Parg]
2.4.0.0	BUGFIX	Core	1	BUGFIX: Core	Synchronous scrape wasn't working [Parg]
2.4.0.0	BUGFIX	Core	1	BUGFIX: Core	Fixed move-on-complete bug when target was a link [Parg]
2.4.0.0	BUGFIX	Core	1	BUGFIX: Core	Fixed DND/Delete vs piece needed/interested in peer issues [MjrTom]
2.4.0.0	BUGFIX	Core	1	BUGFIX: Core	Properly catch Windows shutdown/logoff events [Parg]

2.4.0.0	BUGFIX	UI	1	BUGFIX: UI	Selection moves properly with CTRL+<UP,DOWN,HOME,END> and sorting [TuxPaper]
2.4.0.0	BUGFIX	UI	1	BUGFIX: UI	URL Drag'nDrop Improvements - Unicode, IE Links (Windows) [TuxPaper]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Built in tracker support for multiple listen ports [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Tracker support for multi-hash scrapes [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	"Date added" field added for hosted torrents [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Alternate max upload rate limit when only seeding option [Nolar]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Ability to move a download's data files and torrent added [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Added ability to turn off DHT originated IP Filter violation logs [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	When a peer is banned remove any data downloaded from it in partially complete pieces [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Added "block banning" feature when multiple bad peers found with "close" IPs [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Support for platform-specific plugin update components [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Support for .torrent file download using just infohash hex string (via DHT magnet lookup mechanism) [Nolar]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Added some control over initial share ratio for "add for seeding" downloads [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Added feature to allow disabling of multi-hash tracker scrapes [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Tracker connections respect bind-ip settings (http only) [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Banned ips persisted across restart [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Ability to add an alternative max-torrent-limit when seeding [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	DHT NAT punching for firewalled peers [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Ability to rename and redirect files within a torrent [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Compact storage for "do not download" files [Parg]
2.3.0.6	FEATURE	Core	1	FEATURE: Core	Magnet URI protocol registration under windows [Parg]
2.3.0.6	FEATURE	Plug	1	FEATURE: Plug	UPnP plugin will warn if it discovers a router that has known protocol problems [Parg]
2.3.0.6	FEATURE	Plug	1	FEATURE: Plug	Magnet URI accessor method added to Torrent [Parg]
2.3.0.6	FEATURE	Plug	1	FEATURE: Plug	Added "launchable plugins" that can be used to start Azureus [Parg]
2.3.0.6	FEATURE	Plug	1	FEATURE: Plug	Added single-instance management function [Parg]
2.3.0.6	FEATURE	Plug	1	FEATURE: Plug	Made some platform-specific functions available to plugins [Parg]
2.3.0.6	FEATURE	Plug	1	FEATURE: Plug	Ability to add Swing based interfaces as plugin views [Parg]
2.3.0.6	FEATURE	UI	1	FEATURE: UI	Console UI support for viewing specific

				plugin logs [Parg]
2.3.0.6	FEATURE	UI	1	FEATURE: UI | Beginner configuration user mode provides a simplified interface [Gouss]
2.3.0.6	FEATURE	UI	1	FEATURE: UI | Support for --closedown parameter to org.gudy.azureus2.ui.swt.Main to closedown an existing AZ instance [Parg]
2.3.0.6	FEATURE	UI	1	FEATURE: UI | Added Average Peer Completion percentage column to MyTorrents view and Details view [Nolar]
2.3.0.6	FEATURE	UI	1	FEATURE: UI | Added NAT status icon to status area [Parg]
2.3.0.6	FEATURE	UI	1	FEATURE: UI | Indicate potential DHT port problems in SWT status area [Parg]
2.3.0.6	FEATURE	UI	1	FEATURE: UI | Added swarm average to activity view plus a legend [Parg]
2.3.0.6	FEATURE	UI	1	FEATURE: UI | Console support for listing shares improved, deleting of shares added [Parg]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | Countermeasures against swarm DOS/poisoning [Nolar]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | Health status for swarms where all known peers are connected now reported as OK [Parg]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | Friendly hash-checking option now only applied during recheck operations and delay based on piece size [Parg]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | Don't automatically remove directory contents shares if they (appear to) have been deleted [Parg]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | Improved algorithm for detecting DHT port reachability [Parg]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | DHT - removed cache-distance metrics and added value versions to handle value evolution correctly [Parg]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | Reduced thread count by aggregating async listener dispatchers [Parg]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | Major refactoring of Download and Disk managers [Parg]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | Download totals don't include hash fails and discards and aren't included in share-ratio calculation [Parg]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | "Max simultaneous outbound connection attempts" option no longer accepts zero as a disable value [Nolar]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | Under OSX, new bundle launches using the Java Preferences application configured JVM, enabling JRE 5.0 support [Nolar]
2.3.0.6	CHANGE	Core	1	CHANGE: Core | Under Windows, now runs via a launcher built by exe4j - taskmanager process is now "Azureus.exe" [Nolar,Parg]
2.3.0.6	CHANGE	Plug	1	CHANGE: Plug | Major refactoring of plugin interface to separate out UI-level and core-level stuff [Parg]
2.3.0.6	CHANGE	UI	1	CHANGE: UI | Update to the latest release SWT library (3.1.1) [Nolar]
2.3.0.6	BUGFIX	Core	1	BUGFIX: Core | Fix interested flag being set sometimes when seeding [Nolar]
2.3.0.6	BUGFIX	Core	1	BUGFIX: Core | Wrong file name returned for stopped

				simple torrents [Parg]
2.3.0.6	BUGFIX	Core	1	BUGFIX: Core | Seeder/leecher counts missing for published torrents [Parg]
2.3.0.6	BUGFIX	Core	1	BUGFIX: Core | Stats going negative for hosted torrent average up/down [Parg]
2.3.0.6	BUGFIX	Core	1	BUGFIX: Core | Fix piece request ignore bug [Nolar]
2.3.0.6	BUGFIX	Core	1	BUGFIX: Core | Fixed DHT initialisation hang if port already in use [Parg]
2.3.0.6	BUGFIX	Core	1	BUGFIX: Core | Don't allow prohibited peer sources to be enabled (ui artifact only) [Parg]
2.3.0.6	BUGFIX	Core	1	BUGFIX: Core | Fixed too many socket selectors created when in safe selector mode [Nolar]
2.3.0.6	BUGFIX	Core	1	BUGFIX: Core | Fixed re-announce interval when receiving "failure reason" error response from tracker [Nolar]
2.3.0.6	BUGFIX	Core	1	BUGFIX: Core | Fixed and improved restarts under OSX (and linux) [Nolar,Gouss]
2.3.0.6	BUGFIX	Core	1	BUGFIX: Core | Fixed and improved one-to-one LAN transfer speeds [Nolar]
2.3.0.6	BUGFIX	Plug	1	BUGFIX: Plug | UPnP fix for picking up changed network interfaces [Parg]
2.3.0.6	BUGFIX	UI	1	BUGFIX: UI | Fixed 100% CPU problem with download bars [Parg]
2.3.0.6	BUGFIX	UI	1	BUGFIX: UI | Fixed UI foregrounding itself on every unverified localhost:6880 socket connection attempt [Nolar]
2.3.0.6	BUGFIX	UI	1	BUGFIX: UI | Create-torrent wizard fix for multi-torrent checkbox enabling error [Parg]
2.3.0.4	FEATURE	Core	1	FEATURE: Core | DHT torrent lookup now based on sha1(hash) and torrent xfer encrypted using hash-derived key [Parg]
2.3.0.4	FEATURE	Core	1	FEATURE: Core | More aggressive banning of peers sending bad data - algorithm change and block-banning feature [Gudy,Parg]
2.3.0.4	FEATURE	Core	1	FEATURE: Core | Support for multiple DHT networks on same port [Parg]
2.3.0.4	FEATURE	Core	1	FEATURE: Core | Introduced "passive torrents" - these are tracked but not downloaded [Parg]
2.3.0.4	FEATURE	Core	1	FEATURE: Core | Category setting functions added to "My Tracker" [Parg]
2.3.0.4	FEATURE	Core	1	FEATURE: Core | Added ability to make plugin config ui components invisible [Parg]
2.3.0.4	FEATURE	UI	1	FEATURE: UI | Average peer speed for swarm column added [Nolar]
2.3.0.4	FEATURE	Plug	1	FEATURE: Plug | Plugin interface extension to allow per-plugin torrent-attributes [Parg]
2.3.0.4	FEATURE	Plug	1	FEATURE: Plug | Plugin ResourceDownload feature for accessing content-type [Parg]
2.3.0.4	CHANGE	Core	1	CHANGE: Core | Improved optimistic unchoke anti-leech algorithms [Nolar]
2.3.0.4	CHANGE	Core	1	CHANGE: Core | CVS versions run multiple DHTs to permit validations of changes [Parg]
2.3.0.4	CHANGE	Core	1	CHANGE: Core | DHT anti-spoof for cache forwards [Parg]
2.3.0.4	CHANGE	Core	1	CHANGE: Core | DHT mechanisms for flood prevention [Parg]
2.3.0.4	CHANGE	Core	1	CHANGE: Core | Added local tracker url to torrents when hosting external torrents [Parg]

2.3.0.4	CHANGE	Core	1	CHANGE: Core | Hosted torrents now become "passively tracked" when their corresponding download is removed [Parg]
2.3.0.4	CHANGE	Core	1	CHANGE: Core | More pro-active injection of DHT scrapes for torrents with failing trackers [Parg]
2.3.0.4	CHANGE	Plug	1	CHANGE: Plug | More JPC plugin enhancements to help reduce cache server load [Nolar]
2.3.0.4	BUGFIX	Core	1	BUGFIX: Core | DHT protocol version logic fixes [Parg]
2.3.0.4	BUGFIX	Core	1	BUGFIX: Core | Fix for 100% cpu when more than 60 connections are registered with a selector under buggy network stacks [Nolar]
2.3.0.4	BUGFIX	Core	1	BUGFIX: Core | Invalid listen port configuration would prevent Azureus from starting [Nolar]
2.3.0.4	BUGFIX	Core	1	BUGFIX: Core | Added hooks to catch Windows shutdown events to allow graceful Azureus exit [Nolar]
2.3.0.4	BUGFIX	Core	1	BUGFIX: Core | Fix for start-stopped non-simple torrents showing data-missing error on restart [Parg]
2.3.0.4	BUGFIX	Core	1	BUGFIX: Core | Fix for drag-n-drop SWT exceptions blocking Azureus startup [Nolar]
2.3.0.4	BUGFIX	Core	1	BUGFIX: Core | Fixed up "interested" message for downloads with "do not download" files [Parg,Nolar]
2.3.0.4	BUGFIX	UI	1	BUGFIX: UI | Categories not being correctly displayed when torrents transit between downloading and seeding [Parg]
2.3.0.2	FEATURE	UI	1	FEATURE: UI | Console UI now has update check, alerting and DHT stats [Parg]
2.3.0.2	FEATURE	UI	1	FEATURE: UI | SWT make torrent wizard remembers value for "add other hashes" [Parg]
2.3.0.2	FEATURE	UI	1	FEATURE: UI | Console UI logging config [Fatal]
2.3.0.2	FEATURE	Plug	1	FEATURE: Plug | Added progress indicator and torrent stats to tracker web templates [Parg, Gouss]
2.3.0.2	FEATURE	Plug	1	FEATURE: Plug | Availability column added to webui + some alignment changes [Parg]
2.3.0.2	FEATURE	Plug	1	FEATURE: Plug | XML/http interface access to individual torrent file stats added [Parg]
2.3.0.2	CHANGE	Core	1	CHANGE: Core | CPU usage reductions when connected to many idle peers [Nolar]
2.3.0.2	CHANGE	Core	1	CHANGE: Core | Disable console view logging by default [Nolar]
2.3.0.2	CHANGE	Core	1	CHANGE: Core | Memory usage reductions and optimizations [Gudy, Parg, Nolar]
2.3.0.2	CHANGE	Core	1	CHANGE: Core | Improved long-term connection-attempt management [Nolar]
2.3.0.2	CHANGE	Core	1	CHANGE: Core | DHT bootstrap in absence of version-check server improved [Parg]
2.3.0.2	CHANGE	Core	1	CHANGE: Core | DHT IP filter reports reduced [Parg]
2.3.0.2	CHANGE	Core	1	CHANGE: Core | Disk manager read/write threads now started on demand [Parg]
2.3.0.2	CHANGE	UI	1	CHANGE: UI | Default for "add other hashes" in make

				torrent wizard and sharing config changed to false [Parg]
2.3.0.2	CHANGE	UI	1	CHANGE: UI | Retention of log history removed as taking up to 1MB mem [Parg]
2.3.0.2	CHANGE	UI	1	CHANGE: UI | Added missing spaces back into SWT dock item's tooltip [Parg]
2.3.0.2	CHANGE	UI	1	CHANGE: UI | Restore version number on status bar and add protocol rate to the stats view [Nolar]
2.3.0.2	CHANGE	Plug	1	CHANGE: Plug | JPC plugin refactoring to help reduce cache server load [Nolar]
2.3.0.2	BUGFIX	Core	1	BUGFIX: Core | Fix compatibility with JRE 1.4 series under Win32 due to NIO bug [Nolar]
2.3.0.2	BUGFIX	Core	1	BUGFIX: Core | Ignore peers with these data ports config option didn't work the DHT and PEX obtained peers [Nolar]
2.3.0.2	BUGFIX	Core	1	BUGFIX: Core | DHT IP derivation from contacts fixed [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Client support for the 'trackerid' announce extension [Nolar]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Client support for the 'min interval' announce extension [Nolar]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Added options to disable scrape entirely or just for non-running torrents [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | RSS Feed parse APIs added to plugin interface [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Non-blocking sockets based TCP tracker implementation [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Client identification plugin interface [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Separate protocol payload and overhead stats [Nolar]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Data deletion can now defer to Recycle Bin and Trash under Windows and Mac OS X, respectively [Parg, CrazyAlchemist]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Distributed Database [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Decentralised tracking [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Magnet URI for location of decentralised torrents [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Ability to update built-in plugins independently of the core [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Diagnostic collection for system

				properties and config settings [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | added option to move newly completed seeds to the end of the seeding list, rather than the front [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Support for I2P plugin [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Network selection support (public, i2p, tor) [Parg]
2.3.0.0	FEATURE	Core	1	FEATURE: Core | Inter-client peer exchange [Nolar]
2.3.0.0	FEATURE	UI	1	FEATURE: UI | Option to disable small fonts under OSX [Nolar]
2.3.0.0	FEATURE	UI	1	FEATURE: UI | Console support for aliases [Fatal]
2.3.0.0	FEATURE	UI	1	FEATURE: UI | Torrent export option added to SWT UI [Parg]
2.3.0.0	FEATURE	UI	1	FEATURE: UI | Keyboard shortcut parsing system [CrazyAlchemist]
2.3.0.0	FEATURE	UI	1	FEATURE: UI | System tray menus for setting global upload and download bandwidth limits [CrazyAlchemist]
2.3.0.0	FEATURE	UI	1	FEATURE: UI | Synthesized speech alerts for Mac OS X; Go to Azureus / Preferences to enable it [CrazyAlchemist]
2.3.0.0	FEATURE	UI	1	FEATURE: UI | New file icon for Mac OS X [CrazyAlchemist]
2.3.0.0	FEATURE	Plug	1	FEATURE: Plug | Added local host configuration setting for statusmailer [Parg]
2.3.0.0	FEATURE	Plug	1	FEATURE: Plug | Reworked CSS for tracker web pages to create old and new styles [Gouss]
2.3.0.0	FEATURE	Plug	1	FEATURE: Plug | StartStop Rules: First Priority ignore Rules: 0 Peers and SeedsPeers Ratio [Gouss]
2.3.0.0	FEATURE	Plug	1	FEATURE: Plug | RSS Feed added to tracker web pages [Parg]
2.3.0.0	FEATURE	Plug	1	FEATURE: Plug | Tracker stats added to tracker web pages [Parg]
2.3.0.0	FEATURE	Plug	1	FEATURE: Plug | Generic Messaging API: inter-client message passing [Nolar]
2.3.0.0	FEATURE	Plug	1	FEATURE: Plug | Joltid Peer Cache plugin is now bundled with Azureus [Nolar, Parg, Gudy]
2.3.0.0	FEATURE	Plug	1	FEATURE: Plug | I2P plugin [Parg]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | Smarter re-announce interval handling, especially for lopsided swarms [Nolar]

2.3.0.0	CHANGE	Core	1	CHANGE: Core | Added many new peerid identifications [Nolar, Gouss]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | First piece priority option now also prioritizes last piece of file [Nolar]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | NAT check functions even when downloads have already been started [Nolar]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | Update check can now check via configured proxy [Nolar]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | Rewritten download code: decreased cpu usage and faster speeds in LAN network environments [Nolar]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | Network IP_TOS option now sets required registry setting under Win2K/XP [Parg, Nolar]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | Favor establishing inbound peer connections, to increase the health of swarms with firewalled clients [Nolar]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | Improved seeding unchoking algorithm: distributes data more evenly [Nolar]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | Anti-leech code to prevent optimistic unchoke leechers [Nolar]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | Reduced threads required for multiple torrents [Parg]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | Relaxed cross-torrent file locking to permit multiple read access [Parg]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | changed HTTP user-agent to include OS and java version [Parg]
2.3.0.0	CHANGE	Core	1	CHANGE: Core | max cache size limited to 32M less than VM size [Parg]
2.3.0.0	CHANGE	UI	1	CHANGE: UI | Reintroduced download completion alerts in Mac OS X (use System Preferences / Sound to set the alert sound of choice) [CrazyAlchemist]
2.3.0.0	CHANGE	UI	1	CHANGE: UI | For Mac OS X, contextual menu items no longer carry images for better compliance with Apple Human Interface Guidelines [CrazyAlchemist]
2.3.0.0	CHANGE	UI	1	CHANGE: UI | RFE #1092614: Informational popup messages now auto-close after 5 seconds - unless the message window is closed manually, or if the mouse is over it / details view is open (the timing will be 'reset' then)

				[CrazyAlchemist]
2.3.0.0	CHANGE	UI	1	CHANGE: UI	Miscellaneous cosmetics updates for Mac OS X [CrazyAlchemist]
2.3.0.0	CHANGE	UI	1	CHANGE: UI	Bug #1112278: https:// URLs now autopaste in Open URL window [CrazyAlchemist]
2.3.0.0	CHANGE	UI	1	CHANGE: UI	Main menu bar should now be more streamlined in its arrangement [CrazyAlchemist]
2.3.0.0	CHANGE	UI	1	CHANGE: UI	Console view will now display information logged (for the duration of the application session) when the Console view is closed [CrazyAlchemist]
2.3.0.0	CHANGE	UI	1	CHANGE: UI	ETA and remaining now take into account DHD files [Parg]
2.3.0.0	CHANGE	Plug	1	CHANGE: Plug	More choices in First Priority ShareRatio [Gouss]
2.3.0.0	BUGFIX	Core	1	BUGFIX: Core	IPFilters loaded from static config file weren't working [Parg]
2.3.0.0	BUGFIX	Core	1	BUGFIX: Core	Fix re-announce interval bug when zero peers in swarm [Nolar]
2.3.0.0	BUGFIX	Core	1	BUGFIX: Core	Send uninterested message when complete while using do-not-download feature [Nolar]
2.3.0.0	BUGFIX	Core	1	BUGFIX: Core	Optimistic Connect, when seeding, no longer drops just random connections [Nolar]
2.3.0.0	BUGFIX	Core	1	BUGFIX: Core	Better recovery from corrupt resume data [Parg]
2.3.0.0	BUGFIX	Core	1	BUGFIX: Core	Better handling of invalid torrent save locations [Parg]
2.3.0.0	BUGFIX	Core	1	BUGFIX: Core	fix for SSL (https) problems introduced by changes in JDK 5.0 [Parg]
2.3.0.0	BUGFIX	Core	1	BUGFIX: Core	fix for hash-fails when running with cache + incremental file creation enabled [Parg]
2.3.0.0	BUGFIX	Core	1	BUGFIX: Core	Fix scrape processing stall bug when scrape url was invalid [Nolar]
2.3.0.0	BUGFIX	UI	1	BUGFIX: UI	Corrected save dialog behaviour on Make Torrent Wizard regarding file selection [CrazyAlchemist]
2.3.0.0	BUGFIX	UI	1	BUGFIX: UI	Bug #953619 concerning Mac OS X: Double-clicking .torrent files now activate Azureus with the expected Finder dialogs (Dragging files to the Dock icon will continue to malfunction for the indefinite future) [CrazyAlchemist]

2.3.0.0	BUGFIX	UI	1	BUGFIX: UI | Bug #1120995: Improper validation could allow unreasonably low global upload cap [CrazyAlchemist]
2.3.0.0	BUGFIX	UI	1	BUGFIX: UI | Main window no longer tries to steal focus on opening (Mac OS X) [CrazyAlchemist]
2.3.0.0	BUGFIX	UI	1	BUGFIX: UI | Pop up messages no longer try to steal focus (Mac OS X) [CrazyAlchemist]
2.3.0.0	BUGFIX	UI	1	BUGFIX: UI | Resolved fuzzy dock icon display under Mac OS X if Column Setup is opened [CrazyAlchemist]
2.3.0.0	BUGFIX	UI	1	BUGFIX: UI | Upload caps for individual transfers now display more sensible values if global upload cap is set to unlimited [CrazyAlchemist]
2.3.0.0	BUGFIX	UI	1	BUGFIX: UI | In My Torrents, a category view will no longer prevent its corresponding table view's horizontal scrolling [CrazyAlchemist]
2.3.0.0	BUGFIX	UI	1	BUGFIX: UI | Download bars should no longer disappear when the main window is minimized (Mac OS X) [CrazyAlchemist]
2.3.0.0	BUGFIX	UI	1	BUGFIX: UI | Certain table column contents no longer 'disappear' when a row is selected (Mac OS X) [CrazyAlchemist]
2.3.0.0	BUGFIX	Plug	1	BUGFIX: Plug | StartStop Rules: Autostart Seed Count Only and Prefer Large Swarms removed First Priority [Gouss]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Manual peer upload blocking (when seeding) and kick-banning [Nolar]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Control added to not cache small files (default no cache for < 1MB) [Parg]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Ability to cancel torrent creation processes added [Parg, Gudy]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Ability to cancel sharing process added [Parg]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Ability to pass multiple torrents to Azureus.exe [Parg]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Added configuration items for tracker processing limits [Parg]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Option to open torrent for seeding in create-torrent Wizard [Parg,Gudy]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Auto-rescan of shared resources + addition/deletion of shares accordingly [Parg]

2.2.0.2	FEATURE	Core	1	FEATURE: Core | Persistence of download and tracker stats for shares [Parg]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Advanced network settings: MTU, SO_RCVBUF, SO_SNDBUF, IPTOS [Nolar]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Optimistic Connect: drop inactive connections in order to find better ones [Nolar]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Added ability to apply updates and shutdown Azureus instead of restarting [Parg]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Can now specify that password protected tracker web is only available via HTTPS, HTTP access -> access denied [Parg]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Can now specify comment to be added to share torrents [Parg]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | resume data no longer saved to torrent files, stored in %user-dir%/active instead [Parg]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Plugin support for installing and uninstaling plugins [Parg]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Category support for shares [Parg]
2.2.0.2	FEATURE	Core	1	FEATURE: Core | Apply updates and defer applicaton to later restart/close [Parg/Gudy]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Added 'share' support to console UI [Nolar]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Option to show confirmation dialog on torrent Removal [Nolar]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Added Seed2PeerRatio item to MyTorrents view [Nolar]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Added detailed connection State to peers view [Nolar]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Added Connected Time item to peers view [Nolar]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Option to add torrent downloads silently (without activating main Azureus window) [Nolar]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Added download speed limit column to MyTorrents [Parg]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Telnet UI added [Fatal]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Auto-open stats option added [Parg]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Plugin install/uninstall wizards [Gudy]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Double click on status bar progress area to give details of update/install/uninstall progress [Parg]

2.2.0.2	FEATURE	UI	1	FEATURE: UI | SWT/Console share support for setting category [Parg]
2.2.0.2	FEATURE	UI	1	FEATURE: UI | Pick out links (anything prefixed with "http") in torrent comments in general view [Parg]
2.2.0.2	FEATURE	Plug	1	FEATURE: Plug | Status mailer support for SMTP port, user + password [Parg]
2.2.0.2	FEATURE	Plug	1	FEATURE: Plug | Status mailer support for plain text notifications [Parg]
2.2.0.2	CHANGE	Core	1	CHANGE: Core | Support azureus.install.path parameter, used by new linux launcher script [Nolar]
2.2.0.2	CHANGE	Core	1	CHANGE: Core | Moved config items for tracker client overrides + UDP to tracker client config [Parg]
2.2.0.2	CHANGE	Core	1	CHANGE: Core | Also do dynamic tracker re-announce interval overrides when seeding; uses peer count only [Nolar]
2.2.0.2	CHANGE	Core	1	CHANGE: Core | NAT check url changed to aelitis server [Nolar]
2.2.0.2	CHANGE	Core	1	CHANGE: Core | Reduce tracker re-announce frequency when incoming connections are accepted (unfirewalled) [Nolar]
2.2.0.2	CHANGE	Core	1	CHANGE: Core | Remove old auto-migration code [Nolar]
2.2.0.2	CHANGE	Core	1	CHANGE: Core | Update restart logic now preserves JVM max mem setting for Sun JVMs [Parg]
2.2.0.2	CHANGE	Core	1	CHANGE: Core | Cleanup and optimize choke-unchoke logic [Nolar]
2.2.0.2	CHANGE	Core	1	CHANGE: Core | Added an option to enable the prioritization of most completed Files, default is off [Gudy]
2.2.0.2	CHANGE	Core	1	CHANGE: Core | Default http tracker to disabled [Parg]
2.2.0.2	CHANGE	UI	1	CHANGE: UI | Create torrent wizard now initially defaults torrent save path from "save to" default [Parg]
2.2.0.2	CHANGE	UI	1	CHANGE: UI | Obey "start stopped" config item for dropped torrents [Parg]
2.2.0.2	CHANGE	UI	1	CHANGE: UI | Torrents no longer need to be stopped before removal is allowed [Nolar]
2.2.0.2	CHANGE	UI	1	CHANGE: UI | Graphical view item refresh optimizations [Nolar]
2.2.0.2	CHANGE	UI	1	CHANGE: UI | Change context menu selection for queue/stop/force-start/recheck to be "at least one can be..."

				rather than "all can be..." [Parg]
2.2.0.2	CHANGE	UI	1	CHANGE: UI | More pre-defined speed ranges for queue slot limits [Parg]
2.2.0.2	CHANGE	UI	1	CHANGE: UI | Improve wording of "delete torrent?" prompt [Parg]
2.2.0.2	CHANGE	Plug	1	CHANGE: Plug | UPnP information messages now off by default [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Fix for kernel panics under MacOSX [ej32206, Nolar]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Handle passing of torrents to already running but not fully initialised process better [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Default save dir was defaulting to ON, changed to OFF [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Tidied up UPnP error handing if action missing [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Scrape mult-tracker torrents correctly when download not running [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Upload/download/share ratio frig for newly added seeds made consistent [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Fix for stalls from system clock being set backwards in time [Nolar]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Fixed up bad behaviour on out-of-disk-space falures [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Fixed torrents getting stuck in READY state [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Fix potential connection establishment backlog [Nolar]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Fix potential DOS by timing out stalled connections [Nolar]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Fix bug in re-announce interval override calculation when connection limit has been reached [Nolar]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Fix for hangs when scraping due to slow DNS resolves [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Fix for UPnP failing if trailing spaces in 'action name' [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Fixed bug in choke/unchoke logic when number of connected peers < max upload slots [Nolar]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Updater problems with non-ascii chars in

				user dir (again!) [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Created torrents had name.utf-8 tag in wrong place [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | UDP scrape responses being handled incorrectly [Parg]
2.2.0.2	BUGFIX	Core	1	BUGFIX: Core | Fixed the "high priority" piece-picking for files [Gudy]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Old language files in user dir causing !missing! item texts [Nolar]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix for window state not being remembered between sessions [Nolar]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix for messages window being closed while animated [Gudy]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix for BUG 1059432 : Download bars spawning multiple times when set to auto open [Gudy]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix for BUG 1061538 : /0 error if no pieces [Parg]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix create-torrent wizard to remember paths properly [Parg]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Fixed up opening of torrent files not ending in ".tor" or ".torrent" [Parg]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix Health icons sometimes not refreshing [Nolar]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix bug with Azureus crashing when quitting on OSX if the Stats view has been opened [Gudy]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix the fuzzy Azureus icon bug under OSX [Gudy]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | IPFilter config view slow to display when thousands of ranges are set [Gudy]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Fixed bug with sharing window re-animating even when opened [Gudy]
2.2.0.2	BUGFIX	UI	1	BUGFIX: UI | Fixed bug 1081886 : stats graphics can now be 2000 pixels wide, instead of only 1600 before [Gudy]
2.2.0.2	BUGFIX	Plug	1	BUGFIX: Plug | UPnP plugin support for multi-homed machines improved [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Disk cache [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Ability to set IP type-of-service (TOS) field for outbound data [Nolar]

2.2.0.0	FEATURE	Core	1	FEATURE: Core | Show current upload and download limits in status area [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Locale selection - options to add in all defined encodings + show lax list [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Change the incoming data port without restarting [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Display "warning message" responses in announce replies [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Added option for tracker to perform a connectability check to peers [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Have message aggregation, for bandwidth savings/efficiency [Nolar]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Torrent HTTP urls now accepted as program command line parameter [Nolar]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Option added to disable system tray [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Concurrent hashing for >1 CPU [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Added ability to not outward connect to a defined set of port numbers [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | SOCKS V4, V4a and V5 (no auth/user+password methods) support for outgoing data connections [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Torrents created by Azureus now include .utf-8 fields [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Dynamic piece request queue scaling [Gudy]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Added torrent name to authentication dialog [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Per-torrent upload speed limits [Nolar,Gudy]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Ability to change the sound played on torrent completion added [Parg]
2.2.0.0	FEATURE	Core	1	FEATURE: Core | Added session uptime to stats [Nolar]
2.2.0.0	FEATURE	UI	1	FEATURE: UI | 'Download' menu added with start/stop all + pause/resume options [Parg]
2.2.0.0	FEATURE	UI	1	FEATURE: UI | Table col sort order default option added, thanks to Charnobo! [Parg]
2.2.0.0	FEATURE	UI	1	FEATURE: UI | URL Downloader window now support to

				set-up referrer and saves last used referrers [Parg, Gudy]
2.2.0.0	FEATURE	UI	1	FEATURE: UI | New Statistic page about the disk cache (yeah more CPU consuming graphs) [Gudy]
2.2.0.0	FEATURE	UI	1	FEATURE: UI | Down/Up speed indicators in main view are now double-clickable to open the Stats View [Gudy]
2.2.0.0	FEATURE	UI	1	FEATURE: UI | Added Path and # Remaining Pieces to Files view, Tracker Name to MyTorrents view [Nolar]
2.2.0.0	FEATURE	Plug	1	FEATURE: Plug | webui support for torrent encoding choice [Parg]
2.2.0.0	FEATURE	Plug	1	FEATURE: Plug | Tracker torrent stats available via xml/http interface [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Show time to re-announce/scrape in general view using hh::mm::ss [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Timeout tracker server operations [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Torrent opener will now report errors opening torrents better [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Complete core networking/messaging rewrite [Nolar]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Stop scrape on UDP V2 torrents when download running [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Refactor of DiskManager [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Protocol tweaking for significantly reduced discard rates [Nolar]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Tracker now treats port value of 0 as 'explicitly firewalled' [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Tracker client will revert to "initialised" state after explicit failure from tracker [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Locale selection changed to show more possible encodings [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | XML stats - downloads ordered by downloading/index then seeding/index [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Reworked connection establishment to conform to new WinXP SP2 limits [Nolar]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | When restarting torrents are checked in smallest->largest size [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Reworked the piece picking so that slow peers don't "block" pieces to be completed [Gudy]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | JAR signing for webui (etc) now based on

				"tools.jar" from Sun JDK due to changes with their 1.5 JRE [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | "Pause" command now greyed out if there is nothing to pause [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Create-torrent wizard now remembers comment data on back button + defaults for open/save dirs [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Reduced memory used of ipfilter list to help support large list sizes [Parg]
2.2.0.0	CHANGE	Core	1	CHANGE: Core | Added "*.*" to list of selectable file types in open-torrent dialogs [Parg]
2.2.0.0	CHANGE	UI	1	CHANGE: UI | Added a legend to the Pieces View [Gudy]
2.2.0.0	CHANGE	UI	1	CHANGE: UI | Pieces View now display if the blocks are in the cache or not [Gudy]
2.2.0.0	CHANGE	UI	1	CHANGE: UI | New Tools menu, Configuration --> Options, Plugins root menu, Language selection moved to config [Nolar,Gudy]
2.2.0.0	CHANGE	Plug	1	CHANGE: Plug | UPnP plugin now only uses plugin interfaces [Parg]
2.2.0.0	CHANGE	Plug	1	CHANGE: Plug | Differentiate UDP and TCP mappings in UPnP desc as some routers need this [Parg]
2.2.0.0	CHANGE	Plug	1	CHANGE: Plug | UpdateLanguagePlugin removed, as no longer maintained [Nolar]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | Throw an error message if existing data file length is too large [Nolar]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | Fix new bug with tracker announce every 10s when no peers were connected [Nolar]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | Fix for renaming files across volumes [Parg]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | Torrent auto-import now works with console UI [Nolar]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | Bug when setting upload to unlimited (forced download to unlimited too) [Parg]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | Set downloaded amount correctly when opening a torrent with existing download data [Parg]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | Detect changed file sizes (smaller) when resuming torrents [Parg]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | Random shuffling of multi-tracker torrent

				URLs fixed [Parg]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | Fix occasional 60sec delay on initial tracker announce [Nolar]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | Fix for reading piece data from a too-long file [Nolar/Parg]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | Stop scraping all announce URLs in a torrent when removing torrent [Parg]
2.2.0.0	BUGFIX	Core	1	BUGFIX: Core | File handle closing [Parg/Nolar]
2.2.0.0	BUGFIX	UI	1	BUGFIX: UI | Labels with '&' displaying incorrectly [Parg]
2.2.0.0	BUGFIX	UI	1	BUGFIX: UI | Torrent Downloader Window : retry button now placed correctly [Gudy]
2.2.0.0	BUGFIX	UI	1	BUGFIX: UI | Fix for clicking the plugins directory and open the folder from the Configuration>Plugins view [Gudy]
2.2.0.0	BUGFIX	UI	1	BUGFIX: UI | Fix for animated message windows not being at the right place on dual monitors. [Gudy,Bluelive]
2.2.0.0	BUGFIX	Plug	1	BUGFIX: Plug | Netgear WGT624 was crashing when processing UPnP requests [Parg]
2.2.0.0	BUGFIX	Plug	1	BUGFIX: Plug | Webui 'view' mode wasn't working [Parg]
2.2.0.0	BUGFIX	Plug	1	BUGFIX: Plug | Webui - excessively long status text messing up table [Parg]
2.2.0.0	BUGFIX	Plug	1	BUGFIX: Plug | azplugins - web tracker pages wasn't picking up "show details" option [Parg]
2.2.0.0	BUGFIX	Plug	1	BUGFIX: Plug | webui - fixed bug with re-ordering of columns [Parg]
2.1.0.4	FEATURE	Core	1	FEATURE: Core | Show built-in plugins in config view [Parg]
2.1.0.4	FEATURE	Core	1	FEATURE: Core | Fallback encoding for torrents (a-zA-Z0-9 type encoding with hex chars for others) [Parg]
2.1.0.4	FEATURE	Core	1	FEATURE: Core | Numbers now localized to selected language [TuxPaper]
2.1.0.4	FEATURE	Core	1	FEATURE: Core | More "polite" hash checking...doesn't stress the cpu/system as much [Nolar]
2.1.0.4	FEATURE	Core	1	FEATURE: Core | Tracker seed retention limit added to limit tracker memory on torrents with high seed counts [Parg]
2.1.0.4	FEATURE	Core	1	FEATURE: Core | Show Bad ips/banned ips in ipfilter window + allow clear/reset [Parg]
2.1.0.4	FEATURE	Core	1	FEATURE: Core | Option to start torrents in a stopped state [Parg]
2.1.0.4	FEATURE	Core	1	FEATURE: Core | Torrent removal rules. Initially to handle

				unauthorised torrents + AZ update torrents [Parg]
2.1.0.4	FEATURE	Dev	1	FEATURE: Dev | More features for torrent creation [Parg]
2.1.0.4	FEATURE	UI	1	FEATURE: UI | New peer columns to help track leechers [TuxPaper]
2.1.0.4	FEATURE	Plug	1	FEATURE: Plug | Experimental data upload facility for tracker web pages [Parg]
2.1.0.4	FEATURE	Plug	1	FEATURE: Plug | WebUI support for "host" operation [Parg]
2.1.0.4	FEATURE	Plug	1	FEATURE: Plug | Run the web interface standalone (outside of a browser) [Parg]
2.1.0.4	FEATURE	Plug	1	FEATURE: Plug | Option to keep hold of UPnP port mappings when closing Azureus [Parg]
2.1.0.4	CHANGE	Core	1	CHANGE: Core | More sensible merging of plugin.properties on plugin update [Parg]
2.1.0.4	CHANGE	Core	1	CHANGE: Core | Retuned tracker connect failure retry interval [Nolar]
2.1.0.4	CHANGE	Core	1	CHANGE: Core | Torrents downloaded by URL now named after torrent if not already .torrent [Parg]
2.1.0.4	CHANGE	Core	1	CHANGE: Core | Performance of ipfilter checking improved [Parg]
2.1.0.4	CHANGE	Plug	1	CHANGE: Plug | Green colour used on tracker web pages now more legible [Nolar]
2.1.0.4	BUGFIX	Core	1	BUGFIX: Core | 99.X% / continuous hash fails fix [Parg]
2.1.0.4	BUGFIX	Core	1	BUGFIX: Core | Basic plugin config model parameters not working in non-swt (e.g. console) mode [Parg]
2.1.0.4	BUGFIX	Core	1	BUGFIX: Core | Fix for 'completed' announce event not being sent on occasion [Nolar]
2.1.0.4	BUGFIX	Core	1	BUGFIX: Core | Fix for occassional missing of resume data write -> recheck on start up [Parg]
2.1.0.4	BUGFIX	Core	1	BUGFIX: Core | Bad peers not being detected on hashfail if they contributed all blocks [Parg]
2.1.0.4	BUGFIX	Core	1	BUGFIX: Core | Better handling of "auto import" + default torrent save dir being the same [Parg]
2.1.0.4	BUGFIX	Core	1	BUGFIX: Core | Strip resume data on open of new torrent [Parg]
2.1.0.4	BUGFIX	UI	1	BUGFIX: UI | Fix bug where failed-hashcheck pieces never reappeared in Pieces view [Nolar]
2.1.0.4	BUGFIX	Plug	1	BUGFIX: Plug | Web UI authentication for torrent download not working with default port URLs [Parg]

2.1.0.2	FEATURE	Core	1	FEATURE: Core | Health items now show if a torrent is being shared/published [Parg]
2.1.0.2	FEATURE	Core	1	FEATURE: Core | Creation of self-signed certificates for SSL (etc) via UI [Parg]
2.1.0.2	FEATURE	Core	1	FEATURE: Core | Set download speed per torrent [Parg]
2.1.0.2	FEATURE	Core	1	FEATURE: Core | Universal Plugin and Play (UPnP) support [Parg]
2.1.0.2	FEATURE	Dev	1	FEATURE: Dev | ToolTip access functions for table cells [TuxPaper]
2.1.0.2	FEATURE	Plug	1	FEATURE: Plug | Tracker plugin supports simple category view [Parg]
2.1.0.2	FEATURE	Plug	1	FEATURE: Plug | Webui support for password protected trackers [Parg]
2.1.0.2	FEATURE	Plug	1	FEATURE: Plug | Webui attempts to get access to system clipboard to fix copy/paste issues [Parg]
2.1.0.2	FEATURE	Plug	1	FEATURE: Plug | Plugin interface extensions for basic plugin config [Parg]
2.1.0.2	FEATURE	Plug	1	FEATURE: Plug | External authorisation framework [Parg]
2.1.0.2	FEATURE	Plug	1	FEATURE: Plug | WebUI + XML/HTTP i/f now have separately configurable username/password via Plugin config (*not* plugin.properties) [Parg]
2.1.0.2	FEATURE	Plug	1	FEATURE: Plug | WebUI can now use signed jars, if configured, so that clipboard access doesn't require local config [Parg]
2.1.0.2	FEATURE	UI	1	FEATURE: UI | Better scrape result reporting, including new "Next Tracker Access" column [TuxPaper]
2.1.0.2	FEATURE	UI	1	FEATURE: UI | Tooltip for Health Icon [TuxPaper]
2.1.0.2	FEATURE	UI	1	FEATURE: UI | New language: Japanese [Gouss]
2.1.0.2	CHANGE	Core	1	CHANGE: Core | Core update checker now uses aelitis.com server to grab latest version instead of SF's one [Gudy]
2.1.0.2	CHANGE	Core	1	CHANGE: Core | Default socket write buffer now a more conservative 1460 bytes for smoother uploading [Nolar]
2.1.0.2	CHANGE	Core	1	CHANGE: Core | Central control of global outgoing peer connection rates...i.e. a better SlowConnect [Nolar]
2.1.0.2	CHANGE	Core	1	CHANGE: Core | Unused potential peer connections are cached for later use [Nolar]
2.1.0.2	CHANGE	Core	1	CHANGE: Core | Tracker now doesn't return peer list on "stopped" event [Parg]

2.1.0.2	CHANGE	Plug	1	CHANGE: Plug | Tracker web plugin configuration moved to plugin config from core config [Parg]
2.1.0.2	CHANGE	Plug	1	CHANGE: Plug | Tracker web contexts now have the option to not apply IP Filters - this affects the web plugin and xml/http interface [Parg]
2.1.0.2	CHANGE	UI	1	CHANGE: UI | About Window is closeable by hitting the 'ESC' key [Gudy]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | Fix for Library Paths using single quotes [Parg]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | Fixed shared plugin dir location on OSX [Parg]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | File Open dialogs now correctly remember their last path [Nolar]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | Plugin loader picking up .zip files instead of corresponding .jar files [Parg]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | Don't delete the imported .torrent file if the import dir happens to also be the Save torrent dir [Nolar]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | Torrents leave READY state quicker after other torrents are re-queued [TuxPaper]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | Fix for Fast Resume not working when a 0-byte file exists in the torrent [Nolar]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | Fix for Fast Resume not working when Incremental File Creation is enabled [Nolar]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | Show error message if previously-allocated data cannot be found, instead of re-creating it [Nolar]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | Force a file handle recycle every 50M read so the OS cache clears (Win2k, possibly other OSes) [TuxPaper]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | Tracker stats wrong [Parg]
2.1.0.2	BUGFIX	Core	1	BUGFIX: Core | Not deleting backup torrents (.bak) when MyTorrent->remove + delete [Parg]
2.1.0.2	BUGFIX	UI	1	BUGFIX: UI | # Column now on by default again [TuxPaper]
2.1.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix for Bug #966867: Context Menu not appearing after setting up columns on Linux [TuxPaper]
2.1.0.2	BUGFIX	UI	1	BUGFIX: UI | IP column now sorts by hex groups [TuxPaper]
2.1.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix for .torrent icons with new Azureus.exe

				[Parg]
2.1.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix for text color on some Table cells not moving when row moved [TuxPaper]
2.1.0.2	BUGFIX	UI	1	BUGFIX: UI | Moving torrent from Uncategorized category to new category now removes the row from the display [TuxPaper]
2.1.0.2	BUGFIX	UI	1	BUGFIX: UI | Fix for empty table rows appearing in Peers Details tab and Pieces tab [TuxPaper]
2.1.0.2	BUGFIX	UI	1	BUGFIX: UI | Windows : Tray Icon should come back after explorer crash (needs latest SWT build) [SWT team, Gudy]
2.1.0.2	BUGFIX	UI	1	BUGFIX: UI | 192 bytes memory leak fixed on OSX [SWT Team, Gudy]
2.1.0.0	FEATURE	Core	1	FEATURE: Core | Ability to limit global download speed [Parg]
2.1.0.0	FEATURE	Core	1	FEATURE: Core | Added ability to automatically exclude files when making torrents (e.g. .DS_Store & Thumbs.db) [Parg]
2.1.0.0	FEATURE	Core	1	FEATURE: Core | Caching of peer info to disk for quick restarts if tracker is unavailable [Parg]
2.1.0.0	FEATURE	Core	1	FEATURE: Core | Detection of plugin updates [Parg]
2.1.0.0	FEATURE	Core	1	FEATURE: Core | HTTP scrapes to same tracker combined into one request if tracker supports it [TuxPaper]
2.1.0.0	FEATURE	Core	1	FEATURE: Core | Support for scraping trackers like http://tracker.boxtorrents.com:6969/ that scrape with /scrape but don't have "announce" in announce URL [Parg]
2.1.0.0	FEATURE	Core	1	FEATURE: Core | Tracker connections proxy support (peer connections not supported yet) [Nolar]
2.1.0.0	FEATURE	Core	1	FEATURE: Core | UDP authentication protocol added [Parg]
2.1.0.0	FEATURE	Core	1	FEATURE: Core | UDP tracker version 2 support added [Parg]
2.1.0.0	FEATURE	Core	1	FEATURE: Core | Generic update mechanisms for core, updater and swt [Parg/Gudy]
2.1.0.0	FEATURE	Core	1	FEATURE: Core | Support for loading user-specific plugins from user dir and shared ones from app dir [Parg]
2.1.0.0	FEATURE	Dev	1	FEATURE: Dev | Column management for any of Azureus' table views. [TuxPaper]
2.1.0.0	FEATURE	Dev	1	FEATURE: Dev | Easy to use "basic plugin view": see PluginInterface::getUIManager::getBasicPluginViewModel

				[Parg, Gudy]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | Ability to upload torrents with xml/http interface [Parg]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | All columns sortable and configurable [TuxPaper]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | Added "Remaining", "DLing For" (time) and "Seeding For" (time) columns to "My Torrents" [TuxPaper]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | Added option to auto-update language file from web (Config -> Interface -> Language) [TuxPaper]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | Added option to show transfer rates in bits/sec [Parg]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | In the Details view, the peer's pieces that we already have are shown in a faded color [TuxPaper]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | Linux system tray support [Gudy]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | MyTracker row right-click support for copying torrent URL to clipboard [Parg]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | Right-click menu sorting of columns (Sorting for OSX) [TuxPaper]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | Send text in My Torrents to clipboard [TuxPaper]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | When torrent data is missing, you can change directory via the context menu [TuxPaper]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | OSX : About and Preferences items are listed under 'Azureus' menu [Gudy]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | Added an option not to use units bigger than MB [Gudy]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | Option to show peer host names instead of IP address [Parg]
2.1.0.0	FEATURE	UI	1	FEATURE: UI | Embedded tracker IP blocks shown in Blocked IPs List [Parg]
2.1.0.0	FEATURE	WebUI	1	FEATURE: WebUI| Web Plugin now can set upload rate [Parg]
2.1.0.0	FEATURE	WebUI	1	FEATURE: WebUI| Web Plugin support for uploading torrents [Parg]
2.1.0.0	FEATURE	WebUI	1	FEATURE: WebUI| Webui + xml/http "access" property support for IP range [Parg]
2.1.0.0	FEATURE	WebUI	1	FEATURE: WebUI| Webui + xml/http plugins have had basic plugin view added [Parg]

2.1.0.0	CHANGE		1	CHANGE: | IRC and Tracker Web Pages moved to separate plugin [Parg]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | Auto-imported .torrent files are moved (not copied) to default .torrent save dir if enabled [Nolar]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | Auto-imported .torrent files are renamed *.imported if default .torrent save dir is not enabled [Nolar]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | Can set per-torrent and global peer connection limits [Nolar]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | Config/pref/torrent file saving uses intermediate .saving file for more reliability [Nolar]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | Currently-connected peer connections are dropped when IPFilter is enabled [Nolar]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | File descriptor handles increased from default of 256 to 8192 under OSX [Nolar]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | Ignore Share Ratio can now be non-integer [TuxPaper]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | New SHA-1 hasher: up to 25% faster [Gudy / Nolar]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | Scrape interval now based on # of seeds (15min minimum) [TuxPaper]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | Socket writes now done in full MSS-sized chunks [Nolar]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | Upload limit can now be set less than 5KB/sec. However, doing so limits download speed too [Parg]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | User config/pref/plugins dir culled from Windows' Registry (needs aereg.dll) [Parg]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | OSX user- pref/plugin dir moved from ~/Library/Azureus/ to ~/Library/Application Support/Azureus/ to meet osx standards [Nolar]
2.1.0.0	CHANGE	Core	1	CHANGE: Core | Unix user- pref/plugin dir moved from ~/Azureus/ to ~/.Azureus/ to meet unix standards [Nolar]
2.1.0.0	CHANGE	UI	1	CHANGE: UI | All progress/piece bars re-done (again) [TuxPaper]
2.1.0.0	CHANGE	UI	1	CHANGE: UI | Azureus should work with SWT 2.12 until we break backwards compat. again [TuxPaper]
2.1.0.0	CHANGE	UI	1	CHANGE: UI | Shrink "My Torrents" Context menu [TuxPaper]

2.1.0.0	CHANGE	UI	1	CHANGE: UI | Systray4j removed: system tray support now from SWT built-in code (Requires SWT-M8+) [Gudy]
2.1.0.0	CHANGE	UI	1	CHANGE: UI | Torrent name shown with IPs in the blocked-IPFilter list [Nolar]
2.1.0.0	CHANGE	UI	1	CHANGE: UI | The Donation Window is now Closeable using the 'Esc' Key [Gudy]
2.1.0.0	CHANGE	UI	1	CHANGE: UI | In the Donation Window the OK button should be on top of other Controls [Gudy]
2.1.0.0	CHANGE	UI	1	CHANGE: UI | Added a "what's new" item in help menu, pointing to changelog for current version [Gudy / Gouss]
2.1.0.0	BUGFIX	Core	1	BUGFIX: Core | Files incorrectly shared if contents not a torrent when opening [Parg]
2.1.0.0	BUGFIX	Core	1	BUGFIX: Core | First Priority rules based on time now work across sessions [TuxPaper]
2.1.0.0	BUGFIX	Core	1	BUGFIX: Core | Fix for saving of .torrent file in wrong dir: Bug #916137 [Nolar]
2.1.0.0	BUGFIX	Core	1	BUGFIX: Core | Fix for some discarded data due to occasional duplicate request [Nolar]
2.1.0.0	BUGFIX	Core	1	BUGFIX: Core | Fix for system clock changes stalling downloads: Bug #918193 [Nolar]
2.1.0.0	BUGFIX	Core	1	BUGFIX: Core | Fix for system clock running faster when using Azureus [Nolar]
2.1.0.0	BUGFIX	Core	1	BUGFIX: Core | Fix for underlying socket handles not closing under linux ("Too many open files") [Nolar]
2.1.0.0	BUGFIX	Core	1	BUGFIX: Core | Individual file priorities remembered after Stop-Start [Nolar]
2.1.0.0	BUGFIX	Core	1	BUGFIX: Core | Multiple shares of same resources causing problems (e.g. share contents+share contents recursive of same dir) [Parg]
2.1.0.0	BUGFIX	Core	1	BUGFIX: Core | Simpler and more reliable file allocation...won't b0rk existing data [Nolar]
2.1.0.0	BUGFIX	Core	1	BUGFIX: Core | Re-check on completion no longer sends Have messages [Nolar]
2.1.0.0	BUGFIX	UI	1	BUGFIX: UI | Fix for the General View in a torrent details, not being layout correctly [Gudy]
2.1.0.0	BUGFIX	UI	1	BUGFIX: UI | Fix for the toolbar on linux / OS X [Gudy]
2.1.0.0	BUGFIX	UI	1	BUGFIX: UI | Fix for the Torrent Maker not getting the correct Tracker when choosing from the Combo on OS X

				[Gudy]
2.1.0.0	BUGFIX	UI	1	BUGFIX: UI | Fix for the Freeze on exit under OSX [Gudy]
2.0.8.4	CHANGE		1	CHANGE: WebUI plugin included in mainline (see http://azureus.sf.net/CVS/web.interface.howto.htm)
2.0.8.4	CHANGE		1	CHANGE: Much more reliable SF mirror auto-update handling [Nolar]
2.0.8.4	BUGFIX		1	BUGFIX: Fix for loading .torrent files via web-browser/shell/doubleclick/etc [Nolar]
2.0.8.2	FEATURE		1	FEATURE: Show last time IPFilter list was updated in status area [Parg]
2.0.8.2	FEATURE		1	FEATURE: Support for "compact" tracker announce protocol [Parg]
2.0.8.2	FEATURE		1	FEATURE: Support for "key" tracker announce protocol [Parg]
2.0.8.2	FEATURE		1	FEATURE: Download Speed column in "My Torrents" turns red if below speed set in Queue config. [TuxPaper]
2.0.8.2	FEATURE		1	FEATURE: Webplugin swing ui has status area with total ul/dl [Parg]
2.0.8.2	FEATURE		1	FEATURE: XML over HTTP remote plugin interface (initially to support GTS) - mail parg@users.sf.net for details [Parg]
2.0.8.2	CHANGE		1	CHANGE: All config/pref files utilize .bak backup files [Parg]
2.0.8.2	CHANGE		1	CHANGE: Option to limit outstanding disk writes and piece hash checks [Parg]
2.0.8.2	CHANGE		1	CHANGE: Can delete shares in QUEUED state [Parg]
2.0.8.2	BUGFIX		1	BUGFIX: Fix of startup issues under Win95/98/Me [TuxPaper]
2.0.8.2	BUGFIX		1	BUGFIX: Locale-specific dir creation [Parg]
2.0.8.2	BUGFIX		1	BUGFIX: Hebrew language works now [TuxPaper]
2.0.8.2	BUGFIX		1	BUGFIX: Fixed long load times under Linux of Configuration view and General tab [TuxPaper]
2.0.8.2	BUGFIX		1	BUGFIX: Popup windows now center in main monitor on multi-monitor setups [TuxPaper]
2.0.8.2	BUGFIX		1	BUGFIX: Fix for Pieces and Files views off-by-one row drawing glitch under linux [TuxPaper]
2.0.8.2	BUGFIX		1	BUGFIX: Sort on Health and Availability now enabled [TuxPaper]
2.0.8.2	BUGFIX		1	BUGFIX: Various memory leaks (DiskManager instances not being freed) [Parg]

2.0.8.0	FEATURE		1	FEATURE: QUEUED status. Torrents that are queued are stopped, but available for automatic starting [TuxPaper]
2.0.8.0	FEATURE		1	FEATURE: Partial support for Read-Only data (for seeding) [TuxPaper]
2.0.8.0	FEATURE		1	FEATURE: Added "Forced-Start" to force a torrent to start, ignoring download limits or seeding rules [TuxPaper]
2.0.8.0	FEATURE		1	FEATURE: Auto-positioning of finished torrent based on how badly the torrent needs seeding [TuxPaper]
2.0.8.0	FEATURE		1	FEATURE: Added Availability, Seeding Rank, SavePath, Max # Uploads, and Total Speed columns to My Torrents view [TuxPaper]
2.0.8.0	FEATURE		1	FEATURE: Categories to group your torrents in (right click on My Torrents and select Set Category to get started) [TuxPaper]
2.0.8.0	FEATURE		1	FEATURE: More auto-seeding options [TuxPaper]
2.0.8.0	FEATURE		1	FEATURE: Ability to limit maximum number of file handles open/in-use at any given time [Parg]
2.0.8.0	FEATURE		1	FEATURE: Tracker - various performance enhancements such as announce/scrape caching. Tested to 500,000 peers on single torrent [Parg]
2.0.8.0	FEATURE		1	FEATURE: Tracker activity logging to %azhome%/tracker.log [Parg]
2.0.8.0	FEATURE		1	FEATURE: Tracker bytes in/out + scrapes recorded [Parg]
2.0.8.0	FEATURE		1	FEATURE: Tracker - ability to limit number of peers returned [Parg]
2.0.8.0	FEATURE		1	FEATURE: Share ratio now shown on tracker web pages [Parg]
2.0.8.0	FEATURE		1	FEATURE: Tracker/client support for "no_peer_id" spec for bandwidth savings [Parg/Nolar]
2.0.8.0	FEATURE		1	FEATURE: More plugin stuff - alert raising, ipfilter reloading, various other stuff [Parg]
2.0.8.0	FEATURE		1	FEATURE: Applet UI enhanced into usable state (start/stop/add/remove downloads) + auto refresh [Parg]
2.0.8.0	FEATURE		1	FEATURE: Added torrent hashes for G2 + ED2K (ala http://www.torrentaid.com/) [Parg]
2.0.8.0	FEATURE		1	FEATURE: Added colours to file view to show: grey -> requested; red -> data recently written [Parg]
2.0.8.0	CHANGE		1	CHANGE: Split torrents in My Torrents view into 2 lists:

				Downloading and Seeding (Completed) [TuxPaper]
2.0.8.0	CHANGE		1	CHANGE: STOPPED status now means the torrent never auto-starts [TuxPaper]
2.0.8.0	CHANGE		1	CHANGE: Moved icon from Rank column to Name column [TuxPaper]
2.0.8.0	CHANGE		1	CHANGE: Removed Lock Priority. No longer an issue since priority only gets autochanged once [TuxPaper]
2.0.8.0	CHANGE		1	CHANGE: Remove "Lock Start/Stop" menu option. Stop now means no auto-starting/stopping [TuxPaper]
2.0.8.0	CHANGE		1	CHANGE: Re-design of Configuration view [TuxPaper]
2.0.8.0	CHANGE		1	CHANGE: Language files (MessagesBundle_xx_XX.properties) are read in the following order: Azureus user directory, Azureus application directory, JAR file [TuxPaper]
2.0.8.0	CHANGE		1	CHANGE: Remember sorted column and order for all views [TuxPaper]
2.0.8.0	CHANGE		1	CHANGE: Core performance optimizations and major cpu usage reductions [Nolar]
2.0.8.0	CHANGE		1	CHANGE: User prompt when (1) tracker listens fails (2) incoming server port bind fails [Parg]
2.0.8.0	CHANGE		1	CHANGE: Config files/dirs and plugins now stored in OS user dir (Win: C:\Documents and Settings\username\Application Data\Azureus\, OSX: /Users/username/Library/Azureus/, Linux: /home/username/Azureus/) [Nolar]
2.0.8.0	CHANGE		1	CHANGE: Tracker log now includes date as well as time [Parg]
2.0.8.0	BUGFIX		1	BUGFIX: Once a torrent is complete, and you remove the data, it wil no longer start downloading again when seeding rules are on [TuxPaper]
2.0.8.0	BUGFIX		1	BUGFIX: 1st column of MyTorrents no longer has a gap if there's no icon. (Windows Only Bug) [TuxPaper]
2.0.8.0	BUGFIX		1	BUGFIX: Fix for download stalls at 99% / 100% cpu usage bug introduced in 2070 [Parg/Gudy]
2.0.8.0	BUGFIX		1	BUGFIX: Fix for minimize/close causing the program to disappear from view under OSX [Nolar]
2.0.8.0	BUGFIX		1	BUGFIX: Fix for icon bar buttons not being flat (Bug #890166) [Gudy]

2.0.8.0	BUGFIX		1	BUGFIX: More reliable .config and .torrent file save/load management [Nolar]
2.0.8.0	BUGFIX		1	BUGFIX: Better handling of failures when performing "move on complete" [Parg]
2.0.8.0	BUGFIX		1	BUGFIX: Handle torrent file names with trailing spaces and CR/NL (from Mac) [Parg]
2.0.7.0	FEATURE		1	FEATURE: Under Linux + GTK, added an option to setup a vertical offset to re-align graphics [Gudy]
2.0.7.0	FEATURE		1	FEATURE: SSL tracker client now gives option to import certificate rather than manually doing so via "keytool" [Parg]
2.0.7.0	FEATURE		1	FEATURE: Start All Downloads option on tray icon [Nolar]
2.0.7.0	FEATURE		1	FEATURE: Support for trackers that do not return peerIDs on announce [Parg]
2.0.7.0	FEATURE		1	FEATURE: Sharing - allows files/dirs/dir contents (recursive) to automatically have torrents created for them and torrents hosted
2.0.7.0	FEATURE		1	FEATURE: Tracker allows number of peers returned to be limited [Parg]
2.0.7.0	FEATURE		1	FEATURE: Experimental UDP tracker protocol (see Config->Tracker->Extensions). Azureus client/tracker supports this, as does the XBT tracker (http://sourceforge.net/projects/xbtt/) [Parg]
2.0.7.0	FEATURE		1	FEATURE: Tracker support for GZIP encoding [Parg]
2.0.7.0	FEATURE		1	FEATURE: Tracker support for multiple pages [IAmChrist]
2.0.7.0	FEATURE		1	FEATURE: Tracker pages skinable by placing pages in %azhome%/web [Parg]
2.0.7.0	FEATURE		1	FEATURE: Major extensions to plugin interfaces [Parg]
2.0.7.0	FEATURE		1	FEATURE: Simple Swing based remote admin interface (over SSL+password auth) - email parg@users.sourceforge.net for details [Parg]
2.0.7.0	FEATURE		1	FEATURE: Click on hash in general view to copy hash to clipboard [Parg]
2.0.7.0	CHANGE		1	CHANGE: Azureus now identifies itself via User-Agent in tracker http communications [Nolar]
2.0.7.0	CHANGE		1	CHANGE: Less flickering, on all platforms (tested on both windows and linux+GTK) [Gudy]
2.0.7.0	CHANGE		1	CHANGE: Azureus now responds as Server: Azureus <version> in tracker http comms [Parg]

2.0.7.0	CHANGE		1	CHANGE: Unwritten blocks in Pieces view shown in red [Gijs Overvliet]
2.0.7.0	CHANGE		1	CHANGE: Prioritizing first piece of file(s) now optional in config [Nolar]
2.0.7.0	CHANGE		1	CHANGE: Confirmation on data deletion now optional in config [Nolar]
2.0.7.0	CHANGE		1	CHANGE: More intelligent announce url '&numwant=' handling [Nolar]
2.0.7.0	CHANGE		1	CHANGE: On tracker announce errors, retry interval now uses exponential backoff [Nolar]
2.0.7.0	CHANGE		1	CHANGE: Removed multi-port listening, as shared single port is far superior [Nolar]
2.0.7.0	CHANGE		1	CHANGE: More reliable .config and .torrent file writing [Nolar]
2.0.7.0	CHANGE		1	CHANGE: Tracker scraper now honors a 'flags: min_request_interval' response [TuxPaper]
2.0.7.0	CHANGE		1	CHANGE: Plugins can be initialised from plugin.properties freestanding (not just in .jar file) [Parg]
2.0.7.0	BUGFIX		1	BUGFIX: Problem with tracker not responding with "Connection: close" causing SSL session maintainance with HttpsURLConnection to stuff up and fail clients with "recv fail" [Parg]
2.0.7.0	BUGFIX		1	BUGFIX: Fast resuming with Chinese directories: Bug #869749 [Parg]
2.0.7.0	BUGFIX		1	BUGFIX: System tray icon re-shown after explorer.exe crash (again) [Rele]
2.0.7.0	BUGFIX		1	BUGFIX: Fix for potential memory leaks [Nolar]
2.0.7.0	BUGFIX		1	BUGFIX: Fix for stalled piece writing [Gijs Overvliet]
2.0.7.0	BUGFIX		1	BUGFIX: Fix for .torrent file data not being fully written on shutdown [Parg]
2.0.7.0	BUGFIX		1	BUGFIX: Fix for OutOfMemoryError in DiskManager: Bug #865553 [TuxPaper]
2.0.7.0	BUGFIX		1	BUGFIX: Fix for 'Open a URL' downloading of .torrent file which contains white spaces: Bug #878990 [Parg]
2.0.7.0	BUGFIX		1	BUGFIX: Fix for Fast Resume with Chinese torrents: Bug #878015 [Parg]
2.0.7.0	BUGFIX		1	BUGFIX: Fix for excess thread creation when tracker does not support single-infohash scrapes [Nolar]

2.0.7.0	BUGFIX		1	BUGFIX: Fix for hash checking 0-byte files: Bug #853126 [Nolar]
2.0.7.0	BUGFIX		1	BUGFIX: Fix for opening files via Files tab when using MoveOnComplete: Bugs #860822, #878795 [Nolar]
2.0.7.0	BUGFIX		1	BUGFIX: Fix for 100% cpu usage after network disconnect/reconnect: Bugs #873957, #879498 [Nolar]
2.0.7.0	BUGFIX		1	BUGFIX: Fix for downloaded blocks not being written to disk, due to write-thread stall under high load [Nolar]
2.0.7.0	BUGFIX		1	BUGFIX: Fix for various non-unicode torrent encoding issues regarding resume data [Parg]
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	

2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	FEATURE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	CHANGE		1	
2.0.6.0	BUGFIX		1	
2.0.6.0	BUGFIX		1	
2.0.6.0	BUGFIX		1	
2.0.6.0	BUGFIX		1	
2.0.6.0	BUGFIX		1	
2.0.6.0	BUGFIX		1	
2.0.6.0	BUGFIX		1	
2.0.6.0	BUGFIX		1	
2.0.6.0	BUGFIX		1	
2.0.4.2	FEATURE		1	
2.0.4.2	FEATURE		1	
2.0.4.2	FEATURE		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	

2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	BUGFIX		1	
2.0.4.2	CHANGE		1	
2.0.4.2	CHANGE		1	
2.0.4.2	CHANGE		1	
2.0.4.2	CHANGE		1	
2.0.4.2	CHANGE		1	
2.0.4.2	CHANGE		1	
2.0.4.2	CHANGE		1	
2.0.4.2	CHANGE		1	
2.0.4.2	CHANGE		1	
2.0.4.2	CHANGE		1	
2.0.4.2	CHANGE		1	
2.0.4.0	FEATURE		1	FEATURE: Disable (per torrent) the auto priority setting when seeding [Gudy]
2.0.4.0	FEATURE		1	FEATURE: Disable (per torrent) the rules to start / stop a seeding torrent [Gudy]
2.0.4.0	FEATURE		1	FEATURE: Move files to a directory upon completion [Nolar]
2.0.4.0	FEATURE		1	FEATURE: Slowly establish new connections to peers (for those with internet disconnection issues) [Nolar]
2.0.4.0	FEATURE		1	FEATURE: Bind to local IP address [Nolar]
2.0.4.0	FEATURE		1	FEATURE: Export/import torrent file to/from XML file [Parg]
2.0.4.0	FEATURE		1	FEATURE: Export of runtime statistics to XML file [Parg]
2.0.4.0	FEATURE		1	FEATURE: Hosting of torrents using built-in tracker [Parg]
2.0.4.0	FEATURE		1	FEATURE: Trackers used in the 'make torrent' wizard are now remembered.
2.0.4.0	FEATURE		1	FEATURE: Embedded tracker [Parg]
2.0.4.0	FEATURE		1	FEATURE: Enhanced peer client identification [Nolar]
2.0.4.0	FEATURE		1	FEATURE: Publishing of torrents to tracker (as opposed to hosting them) [Parg]
2.0.4.0	FEATURE		1	FEATURE: Choosable color for the progress bars [Gudy]
2.0.4.0	FEATURE		1	FEATURE: Plugin support [Gudy]
2.0.4.0	FEATURE		1	FEATURE: Open a torrent file without using the default save location [Gudy]
2.0.4.0	FEATURE		1	FEATURE: Added a FAQ and a Donate link in Help menu [Gudy]
2.0.4.0	FEATURE		1	FEATURE: New languages : Brazilian-Portuguese, Czech, Lithuanian [Gouss - Translators]
2.0.4.0	CHANGE		1	CHANGE: Some GUI Changes, so that SWT WinXP theme is better supported [Gudy]
2.0.4.0	CHANGE		1	CHANGE: Saving of torrent files to central dir now optional

				[Nolar]
2.0.4.0	CHANGE		1	CHANGE: Checking after crash now remembers Fast Resume data [Nolar]
2.0.4.0	CHANGE		1	CHANGE: Internal refactorization of core classes [Parg]
2.0.4.0	CHANGE		1	CHANGE: More logging of reasons for connection closed [Gudy]
2.0.4.0	BUGFIX		1	BUGFIX: Fixed snub/unsnub multiple peers on the peers list [Gudy]
2.0.4.0	BUGFIX		1	BUGFIX: Save file dialog now opens as a 'save' dialog, and not 'open' (for OSX) [Gudy]
2.0.4.0	BUGFIX		1	BUGFIX: DNS name resolution caching no longer infinite [Nolar]
2.0.4.0	BUGFIX		1	BUGFIX: Better recovery checking of partially-allocated files [Nolar]
2.0.4.0	BUGFIX		1	BUGFIX: Re-check file(s) integrity after completion [Nolar]
2.0.4.0	BUGFIX		1	BUGFIX: Max torrent file size was limited to 1MB - limit removed [Parg]
2.0.4.0	BUGFIX		1	BUGFIX: Fix for several exceptions thrown during socket reads/writes [Nolar]
2.0.4.0	BUGFIX		1	BUGFIX: Fix for exception thrown when peer sends an invalid bitfield [Nolar]
2.0.4.0	BUGFIX		1	BUGFIX: AZ was reporting cumulative upload/download stats to tracker - should be per session [Parg]
2.0.4.0	BUGFIX		1	BUGFIX: Fix for handling single data files larger than 2GB [Nolar]
2.0.4.0	BUGFIX		1	BUGFIX: Corrected socket writing [Gudy]
2.0.4.0	BUGFIX		1	BUGFIX: Weren't sending "complete" event to tracker on download->seeding transition [Parg]

9.2. Linux Kernel Data

Version	Change	Insert	Deletion	Size	Bugs Introduced
2.4.18.0	1013	75299	18349	806	15
2.4.19.0	3745	549895	162806	4480	9
2.4.20.0	3462	406403	152014	3970	41
2.4.21.0	2963	366643	147765	3670	24
2.4.22.0	3895	487094	322019	4960	72
2.4.23.0	1551	171914	110551	2090	45
2.4.24.0	18	35	16	2	16
2.4.25.0	1774	223743	54469	1740	30
2.4.26.0	672	52087	38026	766	33
2.4.27.0	767	75173	35110	826	29
2.4.28.0	676	31351	15026	467	6
2.4.29.0	769	33838	10768	398	8
2.4.30.0	218	4826	2394	99	4
2.4.31.0	50	1294	459	25	8
2.4.32.0	122	2464	1152	54	6
2.6.0.0.	43	218	117	218	22
2.6.1.0	998	40596	50838	759	78
2.6.2.0	2370	177655	88369	2210	117
2.6.3.0	1906	141924	94321	1950	174
2.6.4.0	3185	192361	143740	2150	125
2.6.5.0	2261	101535	56991	1350	271
2.6.6.0	2642	173046	104312	2350	236
2.6.7.0	3772	175409	157607	2930	291
2.6.8.0	4604	250569	1153188	3520	404
2.6.9.0	4712	260926	131570	3290	370
2.6.10.0	5384	322353	289814	5	367
2.6.11.1	3	8	4	1	39
2.6.11.2	4	10	5	1	8
2.6.11.3	21	72	60	5	10
2.6.11.4	23	74	62	5	13
2.6.11.5	31	102	81	6	13
2.6.11.6	36	146	104	8	26

REFERENCES

1. Fenton, N. E. and M. Neil, “A Critique of Software Defect Prediction Models”, IEEE Transactions On Software Engineering, Vol. 25, No. 5, September 1999.

2. Kutlubay, O. and A. Bener, “A Two-Step Model for Defect Density Estimation Using Decision Tree Learning”, Working Paper, Boğaziçi University, Department of Computer Engineering, 2005.

3. Ceylan, E., O. Kutlubay and A. Bener, “Software Defect Identification Using Machine Learning Techniques”. Proceedings of 32nd Euromicro Conference on Software Engineering and Advanced Applications (Euromicro-SEAA 2006), August 29, 2006.

4. Idri, A. and A. Abran, “Fuzzy Case-Based Reasoning Models for Software Cost Estimation”, Theory and Applications, published by Springer-Verlag, 2002.

5. Boehm, B., “Cost Models for Future Software Life Cycle Processes: COCOMO II”. Annals of Software Engineering: Software Process and Product Measurement, Amsterdam, 1995.

6. Boehm, B. and V. Basili, "Software Defect Reduction Top 10 List," IEEE Computer, IEEE Computer Society, Vol. 34, No. 1, pp. 135-137, January, 2001.

7. Porter, A. and L. Votta, “An Experiment to Assess Different Defect Detection Methods for Software Requirements Inspections”, ICSE, pp. 103-112, 1994.

8. Brilliant, S., J. Knight and N. Leveson, “Analysis of Faults in an N version Software Experiment”, IEEE Transactions on Software Engineering, Vol. 16, No.2, 1990.

9. Ostrand, J, E. Weyuker and R. Bell, “Predicting the Location and Number of Faults in Large Software Systems”, IEEE Transactions on Software Engineering, Vol. 31, No.4.

10. Ragg, T. and R. Schoknecht, “Using Machine Learning for Estimating the Defect Content After an Inspection”, IEEE Transactions on Software Engineering, 2004.

11. Pendharkar, P.C. and G.H. Subramanian, “A Probabilistic Model for Predicting Software Development Effort”, IEEE Transactions on Software Engineering, Vol. 31, No.7.

12. Nagappan, N. and T. Ball, “Use of Relative Code Churn Measures to Predict System Defect Density”. ICSE 2005: 284-292.

13. Kung, C., J. Gao, P. Hsia and F. Wen, “Change Impact Identification in Object Oriented Software Maintenance”. ICSM 1994 202-211.

14. Inoue, K., T. Yamamoto, and M. Matsushita “Ranking Significance of Software Components Based on Use Relations”, IEEE Transactions on Software Engineering, Vol.31, No.3, 2005.

15. Robillard, M. and G. Murphy, “Automatically Inferring Concern Code from Program Investigation Activities”, Proceedings of the 18th International conference on Automated Software Engineering, pp. 225-234, IEEE Computer Society Press, 2005.

16. Godefroid, P., N. Klarlund and S. Koushik, “DART: Directed Automated Random Testing”, PLDI ACM , June 12-15, 2005.

17. Padberg, F., T. Ragg and R. Schoknecht, “Using Machine Learning for Estimating the Defect Content After an Inspection”. IEEE Transactions on Software Engineering, Vol. 30, No. 1., 2004.

18. Nagappan, N. and T. Ball, “Use of Relative Code Churn Measures to Predict System Defect Density”. ICSE’05, St. Louis, ACM Conferences, 2005.

19. Pendharkar, P., G. Subramanian and J. Rodger, “A Probabilistic Model for Predicting Software Development Effort”. IEEE Transactions on Software Engineering, Vol. 31, No. 7, 2005.

20. Johnson, P., “Improving Software Development Management through Software Project Telemetry”, IEEE Software, 2005.

21. Jorgensen, M., “Practical Guidelines for Expert-Judgement-Based Software Effort Estimation”. IEEE Software, May-June, 2005.

22. Beck, K., “Embracing Change with Extreme Programming”. IEEE Computer, October, 1999.

23. Ostrand, T., E. Weyuker and R. Bell, “Predicting the Location and Number of Faults in Large Software Systems”. IEEE Transactions on Software Engineering, Vol. 31, No.4, 2005.

24. Wing, J., “A Specifier’s Introduction to Formal Methods”. IEEE Computer, September, 1990.

25. Bowen, J. and M. Hinchey, “Seven More Myths of Formal Methods”, IEEE Software, July, 1995.

26. Clarke, E. and J. Wing, “Formal Methods: State of the Art and Future Directions”, ACM Computing Surveys, Vol. 28, No.4, 1996

27. Sullivan, K., “Software Assurance by Bounded Exhaustive Testing”. ISSTA’04, July 11-14, Boston, ACM Conferences, 2004

28. Gopal, A., T. Mukhopadhyay and M. Krishnan, “The Impact of Institutional Forces on Software Metrics Programs”. IEEE Transactions on Software Engineering, Vol. 31, No.8, 2005.

29. Yang, J., P. Twohey, D. Engler and M. Musuvathi, “Using Model Checking to Find Serious File System Errors”. *In Proceedings of the Sixth Symposium on Operating Systems Design and Implementation (OSDI), December 2004*.

30. Sullivan, K., W. Griswold, Y. Cai and B. Hallen, “The Structure and Value of Modularity in Software Design”. Proceedings of the Joint International Conference on Software Engineering and ACM SIGSOFT Symposium on the Foundations of Software Engineering, September, 2001.

31. Gregoriades, A. and A. Sutcliffe, “Scenario-based Assessment of Nonfunctional Requirements”. IEEE Transactions on Software Engineering, Vol.31, No.5, 2005.

32. Veanes, M., C. Campbell, W. Grieskamp, W. Schulte and N. Tillmann, “Online Testing with Model Programs”. ESEC-FSE’05, September 5-9, 2005, Lisbon, Portugal, ACM Conferences.

33. U.S. General Accounting Office Report, “Patriot Missile Defense: Software Problem Led to Systems Failure at Dhahran, Saudi Arabia”. February, 1992.

34. Nagappa, N. and T. Ball, “Static Analysis Tools as Early Indicators of Pre-Release Defect Density”. ICSE’05, St. Louis, ACM Conferences, 2005.

35. Dyba, T., “An Empirical Investigation of the Key Factors for Success in Software Process Improvement”. IEEE Transactions on Software Engineering, Vol. 31, No.5, 2005.

36. Mock, M., D. Atkinson, C. Chambers and S. Eggers, “Program Slicing with Dynamic Points-To Sets”. IEEE Transactions on Software Engineering, Vol. 31, No.8, 2005.

37. Royce, W., “Successful Software Management Style: Steering and Balance”, IEEE Software, 2005.

38. Boehm, B. and R. Turner, “Management Challenges to Implementing Agile Processes in Traditional Development Organisations”. IEEE Software, 2005.

39. Mens, T. and T. Tourve, “A Survey of Software Refactoring”. IEEE Transactions on Software Engineering, Vol.30, No:2, 2004.

40. Kalyanaraman, V. and K. Trivedi, “A Comprehensive Model for Software Rejuvenation”, IEEE Transactions on Dependable and Secure Computing, Vol.2 No.2, 2005.

41. “Source Forge Projects”, http://www.sourceforge.net, 2006.

42. “The Linux Kernel Archives”, http://www.kernel.org, 2006.

43. “Linux Kernel Bug Tracker“, http://bugzilla.kernel.org/, 2006.

44. Gokhale, S. and R. Lyu, “A Simulation Approach to Structure-based Software Reliability Analysis”. IEEE Transactions on Software Engineering, Vol. 31, No.8, 2005.

45. Russell, S. and P. Norvig, *Artificial Intelligence A Modern Approach*, New York, Prentice Hall, 2003.

46. Sontag, E.D., "Feedback Stabilization Using Two-hidden-layer Nets", IEEE Transactions on Neural Networks, 3, 981-990, 1992.

47. Bishop, C. M., “Neural Networks for Pattern Recognition”, *Oxford University Press*, 1995.

48. “Neural Nets FAQ“, http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-9.html 2006.

49. Swingler, K., *Applying Neural Networks: A Practical Guide,* London: Academic Press, 2004.

50. Berry, M.J.A., and G. Linoff, *Data Mining Techniques*, NY: John Wiley & Sons, 1997.

51. Sahw, M., “Writing Good Software Engineering Research Papers”. ICSE’02 Conference, 1992.

52. Halstead, M. H., “Elements of Software Science”, *Elsevier North-Holland*, Amstterdam, 1977.

53. Mills, E. E., “Software Metrics”, SEI Curriculum Module SEI-CM-12-1.1, SEI Joint Program Office, Hanscom AFB, MA, December, 1988.

54. Mitchell, T. M., “Machine Learning”, McGraw-Hill and MIT Press, 1997.

55. McCabe, T. J., “A Complexity Measure”, *IEEE Transactions on Software Engineering*, 2, 4, pp.308-320, 1976.