The Classification of Domain Concepts in Object-Oriented Systems

by

Maurice Carey

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2013 by the
Graduate Supervisory Committee:

Charles Colbourn, Co-Chair
James Collofello, Co-Chair
Hasan Davulcu
Hessam Sarjoughian
Jieping Ye

ARIZONA STATE UNIVERSITY

May 2013

ABSTRACT

The complexity of the systems that software engineers build has continuously grown since the inception of the field. What has not changed is the engineers' mental capacity to operate on about seven distinct pieces of information at a time. The widespread use of UML has led to more abstract software design activities, however the same cannot be said for reverse engineering activities. The introduction of abstraction to reverse engineering will allow the engineer to move farther away from the details of the system, increasing his ability to see the role that domain level concepts play in the system. In this thesis, we present a technique that facilitates filtering of classes from existing systems at the source level based on their relationship to concepts in the domain via a classification method using machine learning.

We showed that concepts can be identified using a machine learning classifier based on source level metrics. We developed an Eclipse plugin to assist with the process of manually classifying Java source code, and collecting metrics and classifications into a standard file format. We developed an Eclipse plugin to act as a concept identifier that visually indicates a class as a domain concept or not.

We minimized the size of training sets to ensure a useful approach in practice. This allowed us to determine that a training set of 7.5 to 10% is nearly as effective as a training set representing 50% of the system. We showed that random selection is the most consistent and effective means of selecting a training set. We found that KNN is the most consistent performer among the learning algorithms tested. We determined the optimal feature set for this classification problem.

We discussed two possible structures besides a one to one mapping of domain knowledge to implementation. We showed that classes representing more than one concept are simply concepts at differing levels of abstraction. We also discussed composite concepts representing a domain concept implemented by more than one class. We showed that these composite concepts are difficult to detect because the problem is NP-complete.

DEDICATION

For Gwen.

You are the best wife, sometimes editor, endless source of encouragement, and my best friend.
I could not have completed this without you.
Thank you for being so patient!

For my daughters Lenore and Roselynn.

You are the joy of my life.
There is nothing you can't accomplish if you set your mind to it.

For mom.

Unconditional really means a lot.

For dad.

You taught me to love science. This is the result.

For my sister Dottie.

I never would have started down this path without your help.
Thank you for making sure I got where I needed to be.

ACKNOWLEDGEMENTS

I'd like to thank the following people:

The members of my committee: Dr. Charles Colbourn, Dr. James Collofello, Dr. Hasan Davulcu, Dr. Hessam Sarjoughian, Dr. Jieping Ye for their advise, challenges, insights, and feedback that have helped me to grow to a point where I can complete this work. Dr. Gerald Gannod for guiding me down the right path when this research was just beginning.

Dr. Toni Farley, and Dr. Daniel McClary who were always there to talk through a problem either over a beer or a coffee depending on the circumstances. Dr. Henri Naccache on whose recommendation I spent a summer porting an old Java app he had written to a new system. I learned a lot about how to write great Java code that summer. You are missed. Jason Brown who volunteered to classify Panda. This work was instrumental in validating our results.

Richard Whitehouse who told a freshman class of CS students "if there is something not covered in this class and you want to know it, get a book." Many books and years later that is still the best advise I ever received on advancing my education. Dr. Joseph Urban who taught me two important lessons in one semester. No one likes a jackass in their office, and sometimes you have to become the expert on a topic. Both of these have helped me further my career, and finish this work. Dr. Perry Reinert for giving me the chance to prove I could write code for a living in the real world. This experience helped me to realized the importance of software comprehension, and motivated the development of these ideas.

My colleagues over the years in industry who have taught me invaluable lessons about being pragmatic. Especially: Andy Allen, Robert Barth, Bill Dwyer, Neil Fritz, Preston Lee, David Morgan, Ken Myers, Jeff Nickoloff, Chris Ostler, Chris Tranquill, Willie Wheeler, Andy Will, and Catherine Yackshaw.

Joseph Oder for teaching me the importance of working smart and hard, and how to fish. Without these lessons none that followed would have been as effective.

Finally, Amazon.com for delivering all those books I needed over the years so quickly, and being a great place to work developing software.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

The complexity of the systems that software engineers build has continuously grown. What has not changed is the engineers' ability to deal with a limited number of data at any given time; psychologists tell us that the average person has working memory capacity to operate on about seven distinct pieces of information at a time [40]. While improvements in the field, like the widespread use of UML, has led to more abstract software design activities, the same cannot be said for reverse engineering activities. The well known *concept assignment problem* is still being solved at the *line-by-line* level of analyzing source code. The introduction of abstraction to the problem allows the engineer to move farther away from the details allowing his own limited resources to capture a broader picture of the system, increasing his ability to see the role that domain level concepts play in the system.

Computers have been used for everything from recalculating massive spreadsheets to compiling Java into byte code to typesetting this thesis. However, even classification problems that computers have not traditionally been associated with have become familiar territory for machine learning algorithms. These solutions show up even in the sorting of family photos, a task that has until recently been achieved without automation.

Imagine you would like to see all the photos in your digital library of your daughter taken between the ages of two and three that also include her mother in the photo. The time frame is easily solvable by traditional search; however the question of a photo containing two specific people requires a different approach. Thankfully the technology exists today so that using facial recognition a machine learning algorithm can provide a classification of all of your photos given just a few seconds of compute time and your assistance with training. Of course the resulting set of photos may not be perfect, for example some might be your daughter and her aunt, but the reduction in the size of the set to be considered, the ability to manually indicate errors by tagging

the false positives, and the increases in accuracy that come with further training or tagging of the photos make this a compelling solution to the problem.

Classification can be applied to problems in reverse engineering and can achieve many of the advantages seen in photo classification. Core concepts are classes in the system that are implementations of the domain concepts in an ontology of the system. The alternative, a non-concept class, is a class that only exists to support the implementation of the concept classes. By associating the classes in a system with the domain concepts we can build a filter that allows the engineer to view only those parts of the system that relate to the domain knowledge that is captured in the implementation of those classes. This view can be used to assist in accelerating the engineers understanding and comprehension of the system by hiding those aspects that do not directly relate to the domain.

In this work, we present a technique that facilitates filtering of classes from existing systems at the source level based on their relationship to the core concepts in the domain. Filtering the system in this way allows software engineers to comprehend it faster, since the system is reduced to core concepts. Filtering out non-domain classes also simplifies the visualization of the system. Eliminating non-domain classes from the class diagram of a system reduces the complexity of the diagram. Reduced complexity supports higher comprehension and faster learning of the system. The technique presented involves collecting object-oriented metrics from an existing system that are then used in machine learning methods to create a classifier of the system. The classifier is used to identify classes that are likely related to domain level concepts. This approach can simplify the process of reverse engineering and design recovery, as well as other activities that require a mapping to domain level concepts.

Traditionally, concept identification has been approached in the concept assignment problem [7] as a way of mapping source code to programming concepts. In our work we define concept identification as the process of extracting domain knowledge from an existing software system. A perfect concept identification system would take as input the source to an existing system, and produce as output an ontology of the domain knowledge captured in that system.

From the perspective of forward engineering we can imagine the development process as a series of refinements to the domain ontology that ultimately result in a program that can be executed on a machine. Clearly this is a simplification of the real world process of developing software. In practice, the domain is rarely understood well enough at the beginning of development to create a complete and accurate ontology of the domain; however, over time the ontology could be developed with feedback from the iterative development process. From the perspective of reverse engineering we ask what is the relationship between source code artifacts and the domain ontology. The importance of this question is that in practice we do not follow a perfect deterministic development process, instead we act as humans always do. We make mistakes. We learn from those mistakes. We work overtime to meet deadlines and only capture the newly learned knowledge that is necessary to get the product out the door to our customers. This knowledge is captured in the source of the system. We may have the best intention to capture that knowledge in a more suitable place at a later time, but by the time the heat of the moment has passed even if we stay motivated to record the knowledge we forget a minor detail or maybe a few. Even the knowledge that is recorded is of limited value. The non-formal language in which the knowledge is captured allows ambiguity. What is clear to the writer may not be to the reader. Then, even if perfected, recorded knowledge quickly becomes outdated as the understanding of the problem develops in new directions and as customers request new features. Andy Hunt and Dave Thomas discuss many of these issues in the context of performing maintenance on an existing project [28], particularly that even the most formal of languages, the source code, can be deceptive due to the human element. The programmer decides for example to call a method *readData*, formal language only enforces that he must create a unique identifier within a scope not that the actual functionality of the method is consistent with reading data. Given the human element involved in the development process we need a way of reverse engineering the domain ontology of a system from the only representation of the system that is completely updated with the current knowledge of the system. That representation is the source code of the system itself.

Concept classification allows us to filter the classes in a system to only those that we believe would be a direct translation of a domain concept taken from our hypothetical domain ontology. A little more formally, concept classification is the process of answering the question “Is this class a

member of the set of classes produced from an ontology of the domain knowledge of this system?" for each class in the given system. The research investigation described in this work presents one solution to the problem of producing a concept classification oracle.

From a pragmatic perspective a software developer could use a concept classification tool in order to filter a large system to show only those classes that are of the greatest interest at the domain level. This allows the engineer to learn the domain more effectively. As an example of filtering class diagrams, Figure 1.1 shows the complete class diagram of one package from the Panda system, while Figure 1.2 shows the results of filtering all non-concepts from the diagram.

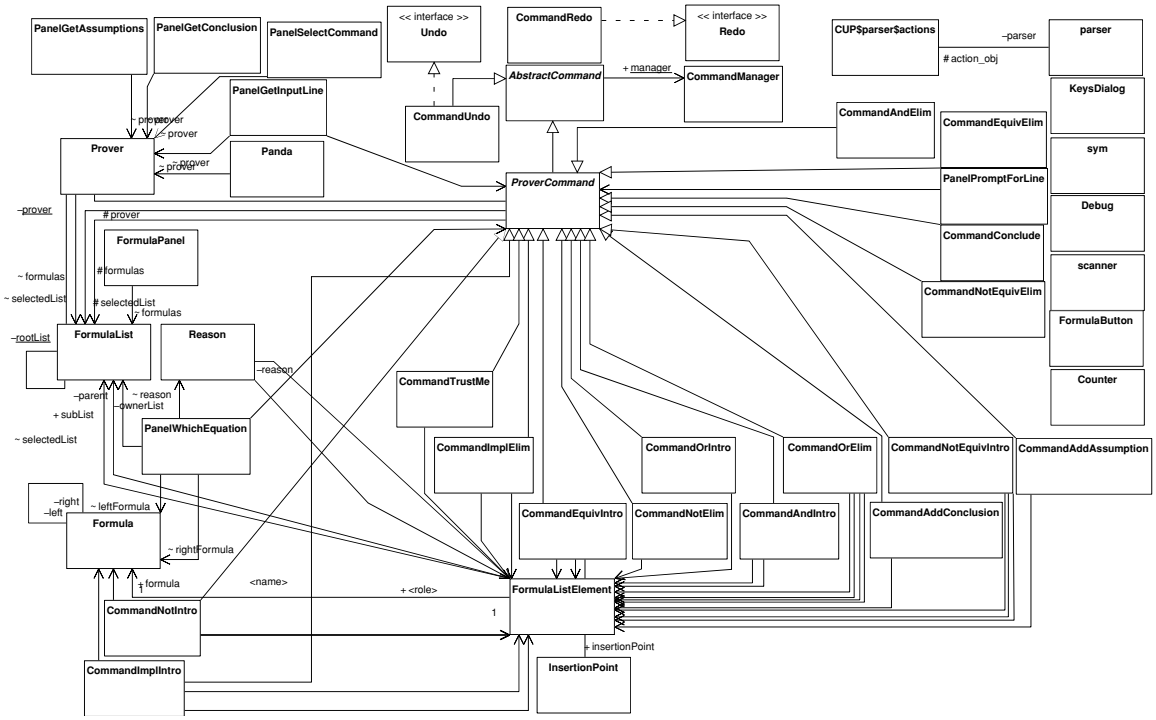

Figure 1.1: Panda class diagram

Other applications of this approach include data mining, and indexing. The open source community has generated a great deal of software over the years. Much of open source software is poorly documented. This issue is further complicated by the transient nature of contributors to open source projects. Not only can our approach help a new contributor to a project learn the domain of that software more effectively, but it can also be used as a way of indexing available

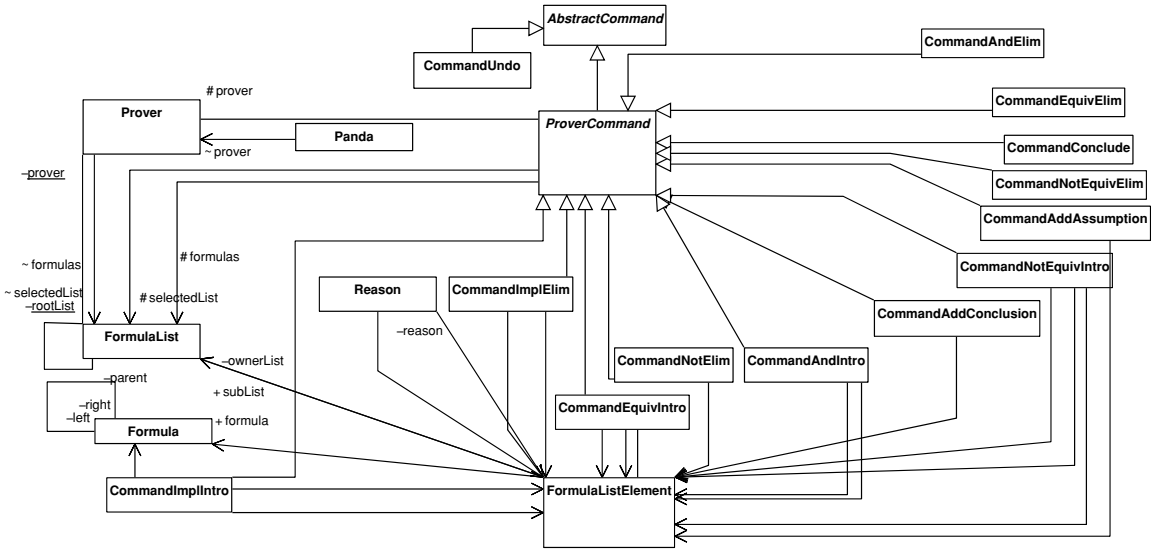

Figure 1.2: Panda class diagram after filtering

open source contributions. Perhaps by providing an index of concepts we can achieve greater reuse of existing software.

1.2 Goals

The following sections present the goals of this research. These goals include attaining a significant level of accuracy in identifying concepts, attaining that accuracy with a small enough training set to be practically useful, optimizing the features that must be collected from the program source, and determining the practicality of generalizing the approach to composite concepts.

Goal 1: Accurate Concept Classification

In Section 3.5, we show statistically significant evidence that the average accuracy achieved on the concept classification problem is better than random guessing or a default learner. A default learner is a simple learning algorithm that "learns" the median value of the training set. For example, if the training set is made up of ten instances and six of those are false then the default learner learns to answer false. In doing so, we address the following questions:

1. Is the average accuracy of a candidate learning algorithm significantly higher than the average accuracy of a default learner on the core concept classification problem?

2. What learning algorithm has the highest accuracy?

Goal 2: Training Efficiency

In Section 4.2, we model the learning curve of the candidate learning algorithms to demonstrate the expected accuracy given a training set of a given size. We define at least one training set selection algorithm, designed such that we are able to build a classifier with higher accuracy than would be possible given a randomly selected training set. In doing so, we address the following questions:

1. How does the accuracy vary with different sized training sets?
2. How large a training set is necessary for a given learning algorithm to achieve significant concept classification accuracy?
3. Is it possible to develop a method of selecting a near ideal training set, where an ideal training set is the training set of a given size which maximizes concept classification accuracy for training sets of that size?
4. Is it possible to develop a method of selecting training sets that consistently produce classifiers with average accuracy above that achieved via random selection?

Goal 3: Feature Selection

In Section 4.1, we optimize the metrics needed to make accurate predictions. In doing so, we address the following questions:

1. What set of metrics is the best indicator for core concept classification?
2. Based on the metrics that are good indicators of core concept classification, are there additional metrics that are useful to collect?

Goal 4: Composition

In Section 5.3, we will explore the use of the approach in classifying core concepts that cross class boundaries. In doing so, we address the following question:

1. Is it possible to use this approach to classify composite concepts made up of more than one class, which individually are not concepts?

1.3 Organization

The rest of the thesis is organized as follows. Chapter 2 contains relevant background information and covers related work in the field. Chapter 3 details the approach and shows that it is possible to attain accurate results. Chapter 4 addresses improvements that can be made in accuracy, practicality of the approach in the real world, and validation of the approach via comparison of another engineers' classification results. Chapter 5 discusses the limitations of our approach as well as classification approaches in general. Finally, chapter 6 discusses the contributions in this work and concludes this thesis.

Chapter 2

Background and Related Work

This thesis makes use of ideas related to several areas of research. In this chapter, we provide the technical background needed later.

2.1 Systems Studied

To evaluate our approach we studied two systems Panda and Scarab. Both systems were completely manually classified in order to generate data sets. These data sets were gathered in order to support our experimental evaluation of the approach. For listings of the data sets collected see Table C.1 on page 90 for Panda and see Table C.2 on page 96 for Scarab in Appendix C.

Panda

Panda [26] is a natural deduction proof assistant written in Java and developed at Arizona State University. The tool is used primarily to support education of natural deduction as a proof technique. We had access to the original class and statechart diagrams that were used for development of Panda and thus used that information as a sanity check during identification of concepts. Panda's size is 90 classes and about 9 thousand lines of code (KLOC).

The architecture of Panda is best described as a GUI based model view controller application. The model consists of classes implementing logical constructs. The view consists of classes implementing Java GUI elements. The controller consists of classes implementing the use cases that can be applied in a proof.

Scarab

Scarab [53] is a web-based defect tracking system based on the Turbine [4] and Torque [3] frameworks. Scarab was designed to be a replacement for defect tracking systems like Bugzilla [12]. The size of Scarab is 580 classes and 128 KLOC.

Turbine is a model view controller framework for web-based applications. Torque is an object-relational mapper for Java classes that allows objects to be persisted to the database. Scarab therefore has the architecture of a MVC web application whose model is persisted to a database.

2.2 The Concept Assignment Problem

Biggerstaff et al. [7] describe the concept assignment problem as recognizing concepts in software systems and building a model or human-level understanding of the concepts. We see the concept assignment problem as a two part process. The first step is to identify concepts in the software system. The second step is to build a model of the concepts. This thesis describes a new method of *identifying* concepts that are represented by classes, and using the identification to produce an abstraction of a recovered class diagram. Specifically, we describe an instance of the concept assignment problem dealing only with object-oriented classes. Object-oriented design suggests that we ignore the details of the implementation of a class, so we believe that analyzing object-oriented software at the class level is a valid approach to solving the concept assignment problem.

2.3 Object-Oriented Metrics

A metric is defined as a standard of measurement [25]. In this work, we use metrics as quantifiable attributes of classes. We are initially more interested in what a set of metrics might say about the class than what each individual metric implies. Since our focus is on attributes of classes, we are most interested in class level metrics. However, we can also make use of lower level metrics that can be averaged over a class.

Metrics Definitions

Several object-oriented metrics are used in our approach. These metrics are primarily designed to capture information about the size and complexity of software at the class level. Section 2.3 provides a brief summary of the metrics used in our work. These metrics were chosen because they were available to be analyzed. We have not optimized the metrics used, nor do we believe that is an appropriate step to take at this time because we do not want to over optimize the metrics used to the data set collected. The best way to view the role of the metrics collected here is as

incomplete indicators of the concepts, each metric potentially adding more information to the decision making process.

Number of Attributes (NOF) The total number of instance and static attributes in the class.

Number of Static Attributes (NSF) The number of class or static attributes in the class.

Number of Methods (NOM) The total number of instance and static methods in a class.

Number of Overridden Methods (NORM) The total number of methods in the class that override methods from an ancestor class.

Number of Static Methods (NSM) The number of static or class methods in the class. Henderson-Sellers [25] refers to this as Number of Class Methods (NCM).

Number of Parameters (PAR) The number of parameters to a method. We do not use this metric directly but instead use the average over the class.

Average Number of Parameters per Class (PARC) The number of parameters to each method in the class averaged over the number of methods in the class calculated as in (2.1).

$$\text{PARC} = \frac{1}{\text{NOM}} \sum_{j=1}^{\text{NOM}} \text{PAR}(j) \tag{2.1}$$

Number of Subclasses (NSC) The total number of direct subclasses of this class, also called Number of Children.

Method Lines of Code (MLOC) The number of lines of code contained in the body of all methods in the class.

Lack of Cohesion of Methods (LCOM) LCOM [24] can be calculated as in (2.2) where $\mu(A_j)$ is the number of methods that access the attribute A_j, a is the number of attributes, and m is the number of methods. LCOM is a measure of the cohesiveness of a class where smaller values indicate more cohesive classes.

$$\text{LCOM} = \frac{\left(\frac{1}{a}\sum_{j=1}^{a}\mu(A_j)\right) - m}{1-m} \tag{2.2}$$

Nested Block Depth (NBD) The depth of nested blocks of code.

McCabe Cyclomatic Complexity (VG) The maximum number of independent circuits through the directed acyclic graph of a method [23]. We use the average over the class.

Weighted Methods per Class (WMC) The sum of McCabe Cyclomatic Complexity for the n methods in the class i, calculated by the formula given in (2.3).

$$\text{WMC} = s_i = \sum_{j=1}^{n} V_{ij}(G) \tag{2.3}$$

Depth of Inheritance Tree (DIT) The depth of the class in the inheritance hierarchy.

Specialization Index (SIX) Defined as $\frac{NORM \cdot DIT}{NOM}$. Designed so that higher values indicate classes that are more specialized.

Uses (USES) The number of classes this class uses, or depends on. This is defined precisely in (2.4).

Used-By (USEDBY) The number of classes that make use of, or depend on, this class. This is defined precisely in (2.5).

$$G \text{ is a directed graph.}$$

$$V(G) = \{v | v \text{ is a class in the system}\}$$

$$E(G) = \{(v_1, v_2) | v_1 \text{ is dependent on } v_2\}$$

$$\text{USES}(G, v) = ||\{e = (v, w) | e \in E(G) \wedge w \in V(G)\}|| \tag{2.4}$$

$$\text{USEDBY}(G, v) = ||\{e = (w, v) | e \in E(G) \wedge w \in V(G)\}|| \tag{2.5}$$

2.4 Machine Learning

The work described makes use of three different machine learning algorithms. Support Vector Machines (SVM) with a radial basis kernel is the first algorithm that we used to classify results. The second, k-nearest neighbors (KNN), is included to compare to other algorithms and is a simple algorithm to implement when contrasted with SVM. The third, ID3 is used as a representative of decision tree algorithms, which have a completely different operational mechanism than either SVM or KNN. We use all the algorithms as a black-box that takes inputs that are vectors of real numbers of length n, and outputs classification decisions. The tool that implements these algorithms and data mining used is known as RapidMiner and formerly YALE [39].

Support Vector Machines

Support Vector Machines (SVM) were first introduced by Boser et al. [8]. SVM are an example of a learning methodology known as supervised learning [16]. A learning methodology is an approach to creating programs that calculate answers by analyzing given examples while supervised learning uses examples consisting of input-output pairs. The goal in a learning problem is to find a function that maps the given inputs to the desired outputs.

SVM have been applied to a diverse body of problems, while many times proving to be the most accurate learning algorithm available [18, 30, 55, 42, 50]. Some examples of problems where SVM have been applied include:

Text Categorization: the classification of text into existing categories. SVM outperformed other machine learning approaches in this area [18, 30, 50].

Hand-written character recognition: the classification of hand-written characters to the categories defined by the set of alphanumerics. This is an interesting problem domain because SVM outperforms algorithms specifically designed for this domain [55].

Image recognition: the classification of objects appearing in computer images. SVM outperformed other algorithms and had the property of being trained on fewer examples than the dimensionality of the data [42].

SVM generate linear functions as their hypotheses. The hypotheses are then applied to attributes that have been transformed to a high dimensional feature space. The transformation of attributes to a feature space is carried out by the application of kernels to the example datum. An example of a kernel is the radial basis function kernel shown in Equation (2.6) [49], which intuitively describes a radius of influence, where $i = 1, \ldots, \ell$, $j = 1, \ldots, \ell$, and σ is a parameter to the kernel that scales the radius of the support vectors influence.

$$K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = \exp\left(-\frac{||\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j||^2}{2\sigma^2}\right). \tag{2.6}$$

SVM used as binary classifiers must solve the optimization problem in Equation (2.7) [16].

$$\begin{aligned} \min_{\vec{\mathbf{w}},b} \quad & \langle \vec{\mathbf{w}} \cdot \vec{\mathbf{w}} \rangle \\ \text{subject to} \quad & y_i \left(\langle \vec{\mathbf{w}} \cdot \vec{\mathbf{x}}_i \rangle + b \right) \geq 1, \\ \text{where} \quad & i = 1, \ldots, \ell \end{aligned} \tag{2.7}$$

Geometrically this translates into finding a linear separating hyperplane in the higher dimensional space. The hyperplane is optimized for maximal margin and defined by the *weight vector* $\vec{\mathbf{w}}$ and *bias b*. In practice the dual representation is maximized, allowing optimization to take place in the kernel space with slack terms introduced along with a penalty parameter *C*. In this context, $\vec{\mathbf{x}}_i \in \mathbb{R}^n$

are the n-dimensional training vectors, $y_i \in \{1,-1\}$ are the classification labels for the training vectors, and ℓ is the number of vectors. The set of classification labels $\{1,-1\}$ correspond to classes that are concepts (1), and those that are implementation details (-1).

In our data set, $n = 14$ so the vector $\vec{\mathbf{x}}_i$ has 14 dimensions, each related to a metric. Examples of some sample vectors from the Panda data set are shown in Table 2.1. The metrics and their ordering are as in Section 2.3.

Table 2.1: Panda data set examples.

i	$\vec{\mathbf{x}}_i$	y_i
1	$< 0,0,3,31,11,0,0,1.75,4,0.444,2.75,0.5,0,1 >$	-1
2	$< 4,0,2,331,92,0,2,1.806,32,0.5,2.556,0.917,0,5 >$	1
3	$< 0,0,8,51,16,0,0,1.071,14,0.862,1.143,0.929,0,1 >$	1
4	$< 1,0,2,24,10,0,2,1.667,2,0.5,3.333,0,0,1 >$	-1

k-Nearest Neighbors

k-Nearest Neighbors (KNN) [41, 2, 1] is the simplest instance-based machine learning method. The algorithm is based on all the training instances being points in an n-dimensional space $\mathbb{R}^n$. Instances are classified by calculating the Euclidean distance to all points in the training set. The Euclidean distance, shown in Equation 2.8, is calculated from $a_r(\vec{\mathbf{x}})$ (e.g., the rth attribute of the learning instance vector $\vec{\mathbf{x}}$).

$$d(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = \sqrt{\sum_{r=1}^{n} \left(a_r(\vec{\mathbf{x}}_i) - a_r(\vec{\mathbf{x}}_j)\right)^2} \tag{2.8}$$

The class of the k nearest points is then used as a simple majority rules vote, the class of the majority is assigned as the class of the point in question. An alternative is to weight the class of the k nearest points based on the multiplicative inverse of their distance from the point in question.

Decision Tree Learning with ID3

Quinlan [47] describes the ID3 algorithm for inductive learning of decision trees. A decision tree is a tree where each non-leaf node represents a test of one attribute of the feature, each branch represents a possible value for the attribute, and each leaf node represents the final classification of an instance. Evaluation of an instance is complete when the leaf node is reached by comparing the

values of the instance to the values of the branches of the decision tree beginning at the root, for each attribute and following the branch for the matching value.

Several decision tree based algorithms [47, 48] have been successful classifiers in many applications, and have been found to yield superior results in text classification [50]. ID3 has a few properties that should be noted. First, it is designed for attributes that take on a discrete and numerable set of values. Second, an interesting susceptibility to noise, non-systematic errors, is encountered when noise is introduced to all attributes of a feature resulting in classification errors that peak around a 40% noise level [47].

Imbalanced Data

Data sets that have a significantly higher number of representatives of one class versus the other class are said to have a between-class imbalance [22]. He and Garcia define between-class imbalance as significant imbalances where the class representation ratio is $100:1$ or greater. Data sets with imbalanced distributions lead to classifiers with imbalanced accuracy. He and Garcia state accuracy imbalances tend to exhibit a majority class with close to 100% accuracy and minority class with 0 to 10% accuracy [22]. The data sets studied in our work tend to exhibit a ratio of $2:1$ between-class imbalance so are unlikely to be affected by these issues.

Kennard-Stone algorithm

The Kennard-Stone algorithm [32] as applied to our work is a method of sorting a data set into a deterministic order. The ordering is defined by the greatest difference between samples; the difference is typically euclidean distance. We also refer to Reverse Kennard-Stone which is simply the output of the Kennard-Stone algorithm in reverse order.

Definition 1 (Kennard-Stone algorithm)**.** consists of the following steps:

1. Find the pair of samples with the greatest difference; this pair is now our list of ordered samples L.
2. Find the difference between all samples not in L and all samples in L.
3. Select the sample with the greatest difference and append to L.
4. Repeat 2 until all samples are in L.

Precision and Recall

Precision and recall allow us to determine the suitability of our results when answering particular questions. For example, when filtering class diagrams we might prefer that we display results that are not truly concepts versus filtering those that are concepts. This means we prefer not to have false negatives. Minimizing false negatives corresponds with a high recall rate for the concept class. In another example, we might use the classifier in an information retrieval role where we ask to see a concept or non-concept class. In this role we might decide that precision is more important as we may not care that all concepts are returned but instead prefer that any concept that is returned is truly a concept.

2.5 Automated Reverse Engineering

The topics covered in this section are works related by their goal of automated reverse engineering.

Software Architecture Reconstruction

Software architecture reconstruction is a recent development that represents a reverse engineering process designed to recover architecture specific information. As Favre [21] points out, software architecture is not well defined. Software architecture can only be defined in terms of the audience to which it is presented. An experienced software engineer should recognize the overloaded use of the term to have slightly differing meanings based on who is the target audience. Obviously, the project manager, technical lead, upper management, customers, fellow engineers with similar experience, the newest addition to the team who finished his degree only a few months ago, as well as other stakeholders all have a different perspective on the software architecture. van Deursen et al. [54] seem to ignore this ambiguity but the approach of view-driven software architecture reconstruction called Symphony is introduced. This thesis essentially codifies best practices that have been observed by the authors in the actual practice of software architecture reconstruction.

Comparison

Placed in the context of software architecture reconstruction and in the vocabulary of Symphony our work would be a type of mapping from a source view to a target view, the target view being a static graph of the core concept classes described by a UML class diagram. That said, the scope of

software architecture reconstruction processes far exceeds the scope of our research. It is not our goal to provide an automated method for complete recovery of a software architecture but instead to automatically build up a view of the domain knowledge embodied in the core-concept filtered class diagrams. Further, our target audience is primarily those with a technical understanding of the software.

Latent Semantic Indexing

Marcus et al. [36, 44, 35, 34, 43] describe a method of using latent semantic indexing to map natural language concepts to the locations of those concepts in source code. This approach allows an engineer familiar with the domain of the software to find usages of concepts in the source code.

Marcus et al. use an information retrieval based approach to locate concepts in the systems' source. Specifically, the approach uses Latent Semantic Indexing (LSI) to uncover semantic similarities between queries and the system source, which is broken up into blocks or modules prior to analysis. Queries can either be user generated, or partially automatically generated queries based on top terms that are related to a user specified term. Queries are compared for similarity to blocks or modules of the system source, which is modularized based on language dependent boundaries. For example, a language like C would be broken into procedures whereas Java might be broken into classes.

Comparison

This approach shares some common goals with ours. One specific instance is concept location for the purpose of easing maintenance of the system and improving developer comprehension. However, the approach assumes developer knowledge of the domain. Our approach attempts to locate all domain concepts in an effort to recover a source based description of the domain. The approaches could be used in a complementary fashion, with our approach locating the classes that represent core concepts in the domain and the latent semantic indexing approach to locate further details within the core classes.

Concept Assignment Problem

Biggerstaff et al. [7, 5, 6] describe the assignment of human oriented concepts to implementation oriented concepts. This process is a matching between preexisting notions about the system or existing domain knowledge to textual representations of the system implementation. This is basically the reverse of our approach. They make use of a concept assignment assistant called DESIRE, which includes tools for analyzing code and constructing alternative views of the code. DESIRE also features a Prolog-based inference engine as well as a knowledge-based pattern recognizer. Together the features of DESIRE allow a developer to analyze the system, and create and collect rules in the inference engine. The rules and patterns collected provide more automated support to the developer over time as the analysis of the system is completed.

Comparison

The concept assignment problem addresses the recovery of programming concepts, whereas we recover domain concepts. In addition to the different goals there are differences in approach as well. The approach used by Biggerstaff et al. uses source code where we use metrics based on the source code. We use a machine learning approach that does not involve declaratively creating an expert system. There is a strong case for the methods used to recognize programming concepts, using at times a filtering of the source code based on the developers domain knowledge. Our approach could be added to the toolset used by Biggerstaff et al. as an additional filtering technique facilitating greater automation than their methods alone. The important point to remember with this work is it assumes significant domain knowledge exists with developers using tools to apply that knowledge. Our work assumes that a developer may not know all domain concepts a priori, but could recognize one if presented with it in the source code.

Forward Engineering and Concept Identification

Svetinovic et al. [52] discuss concept identification from a forward engineering perspective illustrated in several case studies. The claims of misuse or misunderstanding of object-oriented domain analysis are worth noting since the automated identification of concepts requires that those concepts be represented within the implementation, implying they where designed into the

software. They identify many of the concerns we have raised in our discussion of external validity. Primarily, there is not one single agreed upon way of designating what is and is not a concept in a software system, according to their paper this is a fundamental difficulty with object-oriented domain analysis.

Comparison

There is no direct comparison to our work as the authors are attempting to illustrate a weakness of the object-oriented domain analysis process. However, if the concepts that are found to be missing in a domain analysis are also found to be missing in object-oriented design and later implementation then it would not be possible for us to recover or identify a concept that is not present in the system. The authors do state that the findings on OODA do not necessarily extend to OOD or later implementations of the systems. It is possible that concepts that are ignored or misunderstood at the time of analysis are later found or corrected during design or implementation.

Machine Learning Approach to the Concept Assignment Problem

Merlo et al. [38] describe a design recovery process which uses a machine learning approach to link text from source code to concepts identified by a domain analysis. This is essentially a partial automation of a portion of the work performed by Biggerstaff et al. [7]. The machine learning algorithm used is based on neural networks.

Comparison

The approach differs from ours in that a change in domain requires a new neural net to be trained. That is not necessarily the case with our SVMs as they can be applied to programs from very different domains but currently without significant accuracy.

Ontology Recovery by User Interface Mapping

Hsi et al. [27] approach recovery of ontology by manual construction of an *interface map*. The interface map is a depth first traversal of all elements of the softwares interface. The approach then uses the interface map to generate a semantic network. The semantic network is then used as the basis for calculating graph theoretic measures that are used as indicators of *core concepts* of the

ontology. The measures or metrics are made use of on an individual basis and no attempt to combine metrics is made.

Comparison

The approach differs from our work in that it is completely based on construction and analysis of a graph. Our work does make use of graph metrics and could easily be extended to include additional metrics. Their evaluation does not include any comparison to a *control group* in order to express the accuracy of the approach. Each of the various methods introduced to recover the ontology produce differing ontologies, and no method of reconciling differing results is presented. Advantages of our approach include the use of a control group in the form of the manual classification results to show the accuracy of the approach, and we only produce one ontology based on the metrics that we use. That said, the metrics used in the paper do appear to show strong evidence of core concepts.

Webmining Techniques in Source Code

Zaidman et al. [58] show that webmining techniques can be applied to source code to uncover important classes, similar in nature to our core concept classes. This paper demonstrates the use of webmining techniques originally designed to identify important hubs on the web. Web hubs are those sites that provide links to authorities on a topic. The technique uses an execution trace to uncover important or core classes in the system. A log of execution is converted into a compact call graph where each vertex is a class and a directed edge is weighted with the number of calls to methods on the class. The HITS algorithm is used to find the hubs and authorities in the graph. Hubs in this case are considered important classes in the system to enable program comprehension.

Comparison

The goal of this approach, to find the most important classes in a system, is similar to ours. This approach requires defining execution scenarios, ours does not. It is not clear how many scenarios would be required to uncover all important classes in the system. Based on the published findings it appears we achieve similar accuracy, using a static approach. Rather than viewing this as a competing approach it would be far more interesting to use the results as a metric in one of our

features. So this could very well be a complementary approach, if the user is willing to create execution scenarios.

2.6 Using Machine Learning in Software Testing

The works described in this section are related by their use of machine learning techniques but primarily describe approaches to improving software testing.

Program Verification

Bowring et al. [9] present an approach that classifies program behaviors based on similarities of Markov models built from the event transitions collected from program execution traces. An event transition is defined as "a transition from one program entity to another". Here an example of a program entity is a source and sink node in a control flow graph when the transition is a branch. The paper presents a method of collecting behaviors with unknown classification that can be compared to known behaviors in order to decide if these behaviors are "pass" or "fail" behaviors. In other words, the software is analyzed to collect a set of execution traces that lead to both passing and failing tests, for known errors. These are then used as a baseline of comparison to other execution scenarios in order to classify those scenarios as either passing or failing to detect latent errors.

Comparison

The goal of this work, to classify the program behavior, differs from our goal of locating concepts. What is significant is the use of machine learning in order to classify behaviors of the system. There is a similarity in the approach of applying machine learning to software engineering activities, which we believe helps prove the usefulness of applying these techniques to software engineering problems. Specifically, machine learning is used as a tool to augment the knowledge of the engineer by creating classifiers that can potentially detect similar behavior in code that was not directly analyzed by the engineer. This is exactly the way we use machine learning to augment the engineer's ability to detect core concepts of the domain knowledge.

Error Detection

Brun and Ernst [11] present an approach designed to uncover latent errors using machine learning. Errors are learned by presenting a machine learner with runtime properties that are converted into

feature vectors for two separate programs, one with errors and one with the errors corrected. It is not necessary to correct all errors in the program. This data allows the machine learning algorithm to learn a characterization of code that leads to error conditions.

Comparison

Though the goal of our work is very different from this paper, the spirit of the approach is similar in the application of machine learning. The premise that it is possible to characterize code based on source which is transformed into a mathematical vector representation of the code is similar to our collection of metrics based on source size and complexity but obviously with a different transformation. In fact, the similarities would seem to warrant further investigation. It appears that many types of classifiers could be generated from source given an appropriate transformation of the source to a feature vector.

Chapter 3

Accurate Concept Classification

In this chapter we discuss the possibility of accurate concept classification using machine learning. We show conclusive statistical evidence that it is possible to detect concept classifications using machine learning. In the following chapters we determine if this is practical and what limitations exist with an approach of this type.

3.1 Vision and Hypothesis

The ability to automatically classify the classes that make up a system as either core domain or non-core concepts is a key advantage to the software engineer. This information can be used to quickly filter the class diagram of the system and see a view of the most important classes in the system. The filtered class diagram can be further refined by application of techniques developed by Egyed [20] and supported by tools, like the one developed by Kaynak [31]. This allows the relationships between the filtered classes to be built despite the missing classes that have been filtered out of the system.

This new view of the system provides the software engineer with a simplified perspective of a complex system. The simplification makes it easier for the engineer to:

- capture a snapshot of the state of the architecture,
- learn the primary system interactions, and
- discover the important components of an unfamiliar system.

These technology innovations allow the engineer to work at a faster pace with larger, more complex systems.

To support this vision we have a simple informal hypothesis. We believe that there exists a set of metrics that capture properties of a system that can be used by a machine to classify concepts of that system. In this chapter we show this is true by demonstrating that a classifier can perform significantly better than random chance. We purposefully do not limit the metrics used in this set

of experiments so as to eliminate any bias on our part from interfering with potential unexpected discovery. Instead we train using all the metrics we had implementations available for at the time, as given in Section 2.3.

3.2 Methodology

Our approach consists of several steps to generate a classifier that can automatically identify classes as either core or non-core concepts in a system. Figure 3.1 shows an outline of the steps in creating a classifier, and Figure 3.2 shows an outline of the process of using the classifier.

1. The analyst must supply a training set in order for a learning algorithm to generate a classifier. This training set is partially provided by the Manual Classifications as seen in Figure 3.1.
2. A set of metrics is collected from the subject system. This is shown in Figure 3.1 as System Metrics Data, which along with Manual Classifications makes up the training set for the learning algorithm.
3. A learning algorithm is used to produce a classifier from the collected metrics and the training set provided by the analyst. This classifier is shown in both Figure 3.1 as the output of the learning algorithm and Figure 3.2 as one of the inputs to the learning algorithm.
4. The classifier is then used to provide classifications for the classes in the system that have not been manually classified. This is shown in Figure 3.2 as the output of the learning algorithm.

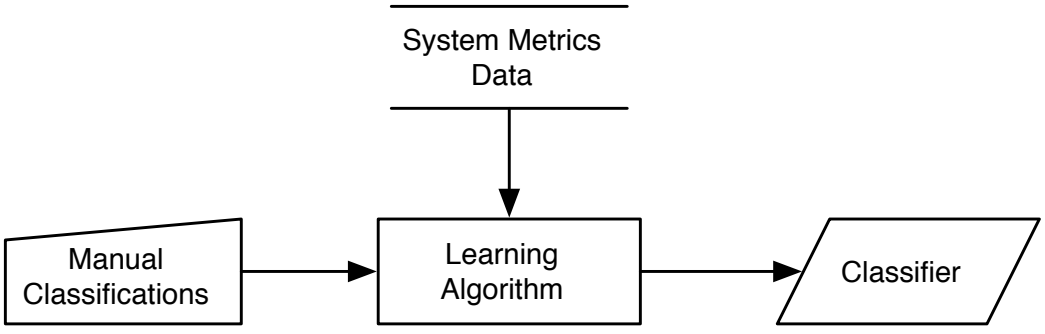

Figure 3.1: Creating a classifier.

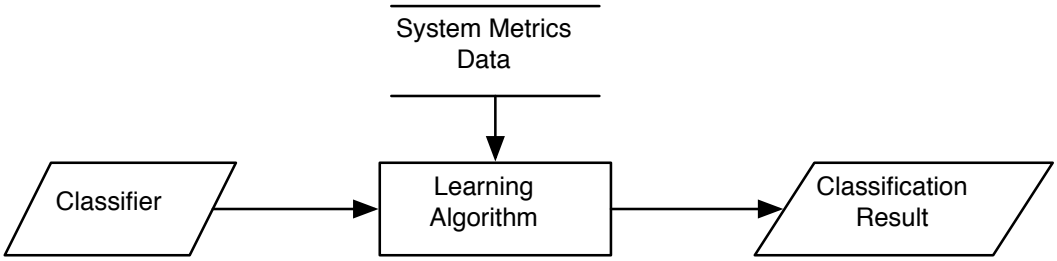

Figure 3.2: Using a classifier.

The result of this process is the complete classification of a system that can be used to filter or otherwise operate upon class diagrams, source, etc. The following subsections describe the details of each of these steps.

Manual Classification

Manual classification is the process of capturing expert analyst knowledge in a machine readable format. This machine readable format can be any representation that shows the mapping between a class and the given classification. Our toolset uses Java annotations to markup the actual program source, but an external database could be used in its place. An analyst completing this task must answer two questions.

1. What is the best class to classify next?
2. What is the classification of this class?

There is potential to provide some automated support for the first question by application of a heuristic to calculate potential information gain from each of the unclassified classes in the system. This is discussed in Section 3.2. The second question is a bit harder to answer and mostly depends on the experience and expertise of the analyst. We discuss potential approaches in Section 3.2.

Selecting the Training Set

There are several possible approaches to the selection of what should be in the training set. The training set could be selected randomly, it could be selected by greatest difference among members, or it could be selected based on the least difference among members. We are not limited

to these techniques alone, however we focus on these in this work as a starting point for the exploration of training set selection for this problem.

Techniques for Classification

While it is not possible to give a complete description of the precise decision making process that occurs when classifying the system, we discuss the general techniques used on the systems we have classified. If we could describe the process of arriving at these decisions then the problem of classifying systems would be solved.

First, we can make use of the observation of Ivar Jacobson, in *Object-Oriented Software Engineering* [29], that the objects making up a system can be divided into three stereotypes including *entity*, *control*, and *boundary*. *Entity* objects are those that are most closely related to the core object concepts of the domain. *Control* objects are most closely related to use cases or actions within that domain. *Boundary* objects mostly consist of the implementation details we would classify as non-core concepts in the domain, however there can be exceptions to this heuristic.

Second, we need to recognize that not all software is designed equally. In the ideal case we are presented with a system where each domain concept is perfectly encapsulated in a single class representation which has crisp separation of concerns with both other concepts and implementation details in the system. On the other end of the spectrum we have a single class that implements everything, meaning this is either a very simple domain or the worst design possibility. Most systems lie somewhere between the two extremes and we hope closer to the former. In practice what this means is sometimes classes represent both a core domain concept, and some implementation level detail we don't care much about. However, this does not defeat our technique as the classification system can still identify these concepts mixed with implementation as long as we provide examples. The important thing we must recognize is that we are not pursuing a theoretical framework to identify precisely a concept. We are trying to identify the concepts as they are presented in the imperfect software even if they include some extra implementation details.

Metrics Collection

Metrics collection is the process of analyzing program source to calculate the desired metrics. For the most part metrics are calculated from the AST of the source code. This process can be completely automated and takes advantage of the rich set of tools that exist in the domain of compilers. Each metric collected can be written as a set of operations on the AST of a class, or a set of ASTs for several classes. For more information on the metrics used see Chapter 2 Section 2.3.

Generating a Classifier

A classifier is generated using a machine learning algorithm. Machine learning algorithms vary a great deal, however for the supervised learning methods we consider, the inputs can be roughly divided into two sets. First, we have the training set that consists of a set of feature vectors made up of the metrics we have collected as well as the classification of the given class. Second, we have some set of parameters to the learning algorithm that varies depending on what algorithm is selected. Regardless of what learning algorithm is used the inputs plus the algorithm give us a classifier that may be expressed in various ways by different algorithms but can be used to classify further examples taken from the system.

Using the Classifier

Once the classifier is generated it can be used to provide a classification for any class for which the same set of metrics can be calculated. Though this would include classes from other object-oriented systems, we limit our use of the classifier to classes in the same system as those used to build the classifier. The primary reasoning behind this limitation is we have not yet found evidence that the classifier is generalizable across domain boundaries [13].

3.3 Tool Support

We have created a tool set that allows easy markup of the system source with annotations, assisting in the manual classification step of the process. The tool, which is implemented as an Eclipse [19] plugin, collects metrics from the source and generates an SVM classifier, and then shows classification results to the user. Figure 3.3 shows an overview of the tool chain process for

generating classifications. The following sections provide a description of the tool features that support each of the activities in our approach.

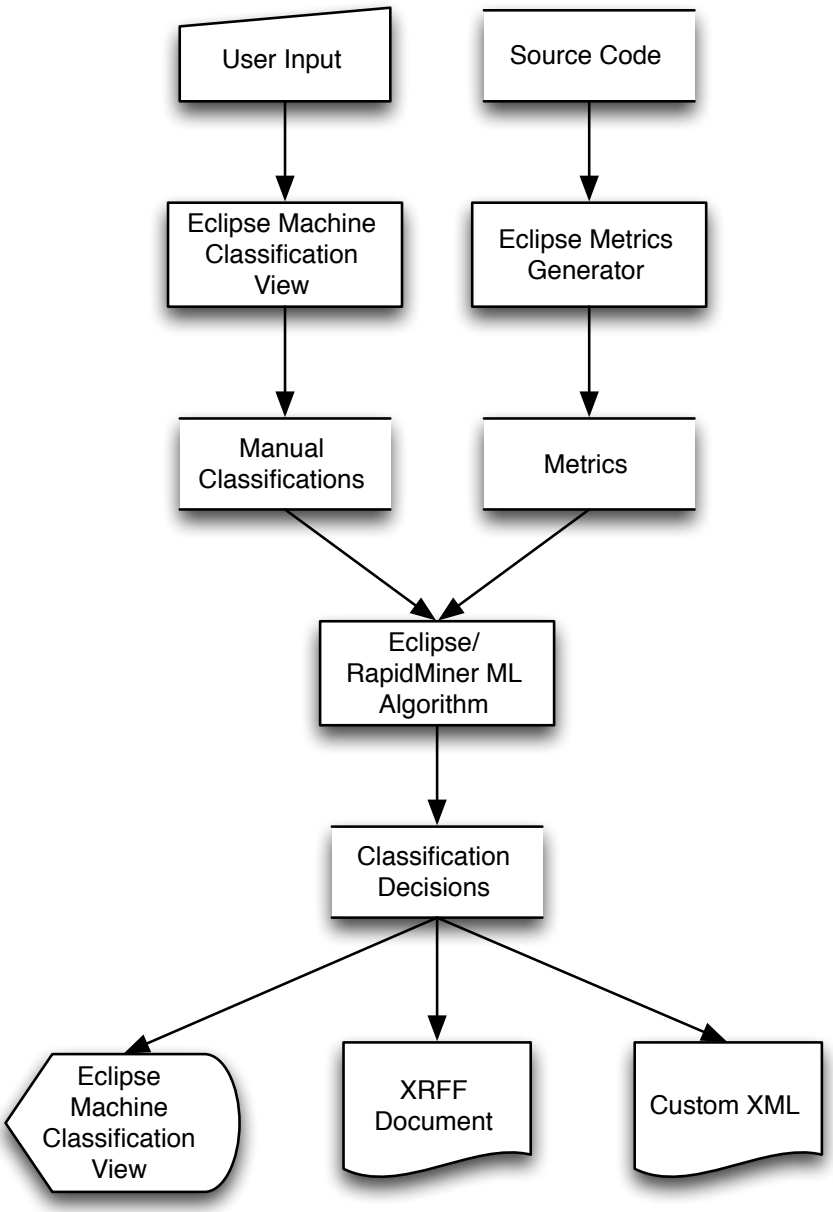

Figure 3.3: Overview of tool chain.

Tool Support for Manual Classification

In order to support the process of manual classification, we have added a simple plugin to Eclipse that allows annotations to be inserted into Java source files in a single step. The interface consists

of two Eclipse actions in a new Eclipse view called the “Machine Classification View”, which can be seen in Figure 3.4. The first action classifies the currently selected class as a core concept, this inserts the complete annotation necessary to support the remainder of the classification process. The second action classifies the currently selected class as a non-core concept. Together these two actions ease the process of classifying classes by replacing a significant amount of typing with a simple click. The possibility of introducing mistakes is also lessened a great deal.

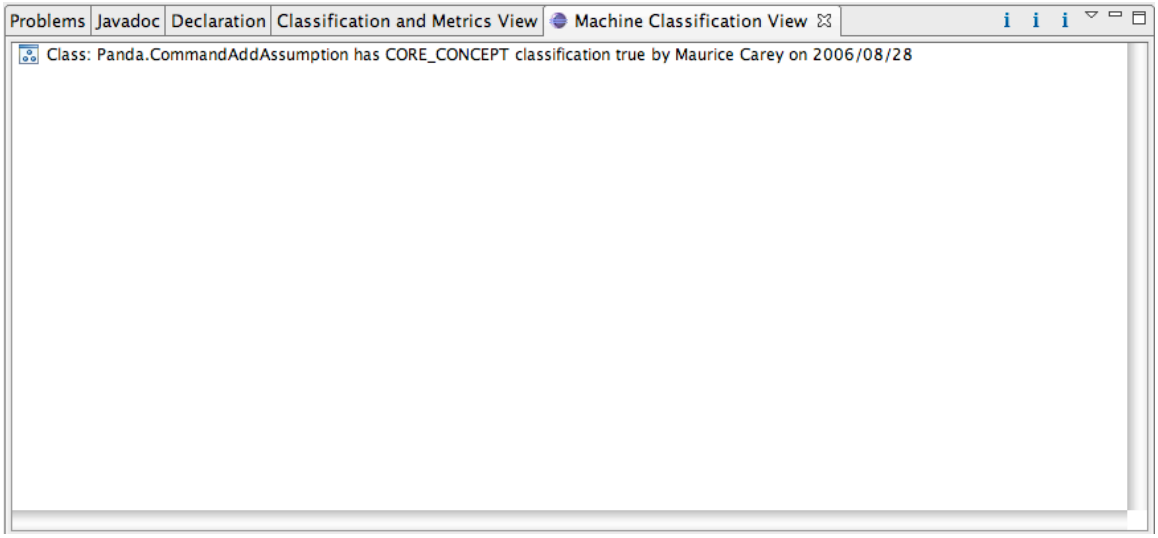

Figure 3.4: Buttons for quick annotations of source

Tool Support for Metrics Collection

To support generation of the metrics data required for classification we have implemented a method to automatically calculate measurements using the Abstract Syntax Tree (AST) included with the Java Development Tool (JDT) plugin in Eclipse. The metrics can be viewed in the new Eclipse view “Classification and Metrics View” that is shown in Figure 3.5. In addition to viewing the metrics associated with a class this view also displays the current classification as defined by the annotation of the class.

Functionality has also been added to allow the export of the classification and metrics to a file. Currently, there is support for a custom XML format file and a eXtensible attribute-Relation File Format (XRFF) [45] file. XRFF is an XML version of the Attribute-Relation File Format

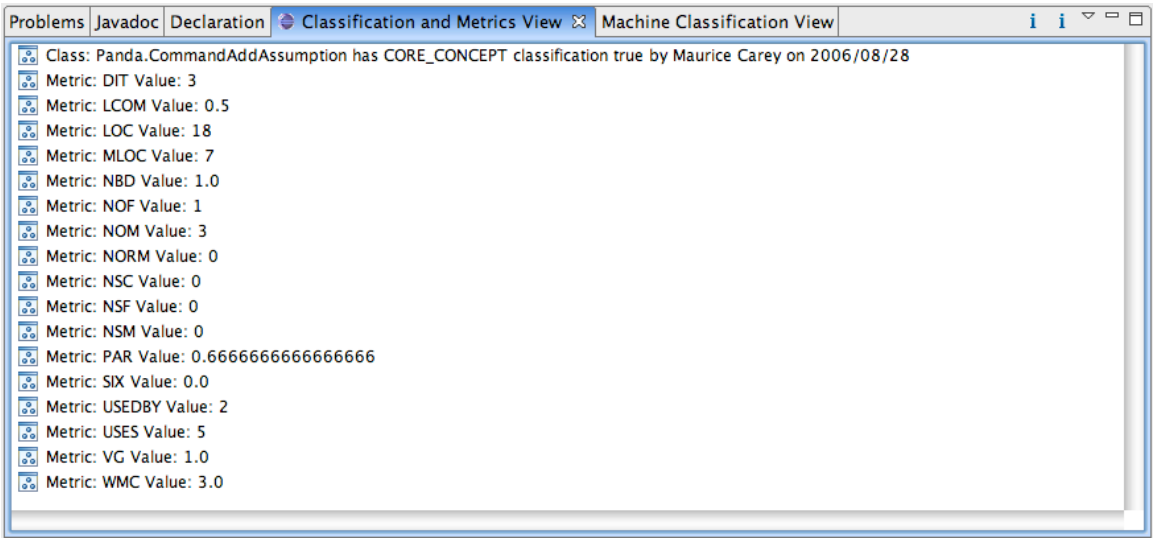

Figure 3.5: Interface for showing metrics output

(ARFF) [46], which is designed to capture the data-sets necessary for performing machine learning experimentation. The export features are not necessary for the methodology defined in our approach but are designed to support evaluation of the approach through experimentation using external toolsets.

Tool Support for Generating a Classifier

A classifier can be generated using the Machine Classification View's "Create SVM Learner" feature, which runs the SVM machine learning algorithm in order to generate a kernel used for further classifications. The training set is selected as all classes currently selected in the Eclipse UI that have annotations applied. This feature is primarily useful in the current experimental phase of the tool, where we typically have classified the entire system and want to perform comparisons of the machine generated classification and the manual classification. Currently, SVM is the only classifier implemented but the framework supports the addition of other classifiers. At the time of implementation of this tool we had seen the highest accuracy using SVM hence the decision on classifier implementation. However, now we know other algorithms can exceed the performance of SVM. This choice does not effect our ability to evaluate the performance of other machine learning algorithms using external toolsets, and was primarily implemented to demonstrate feasibility of the approach.

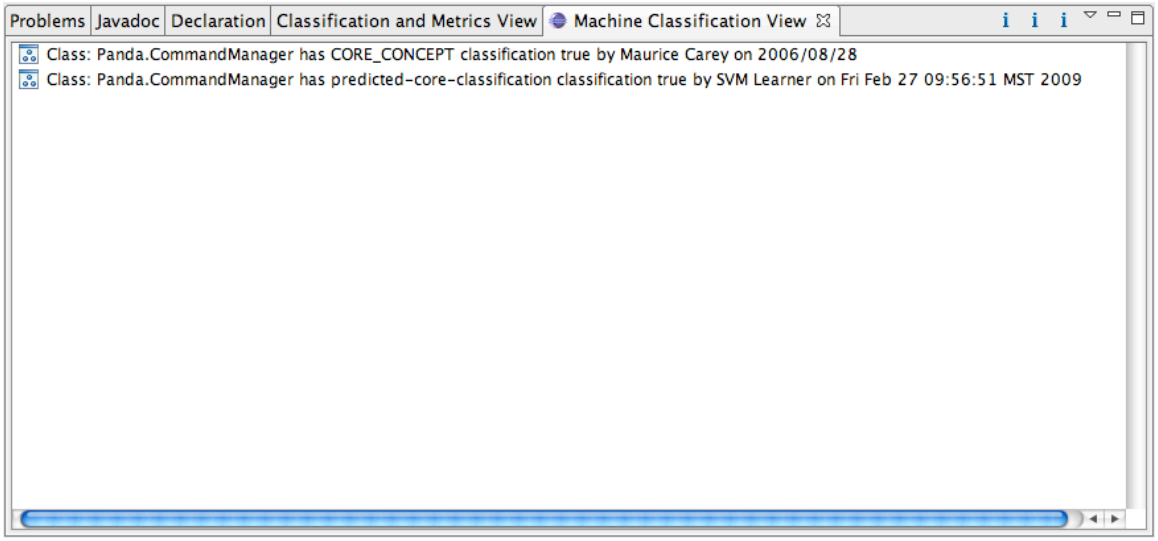

Figure 3.6: Interface for generating a classifier

Tool Support for Using the Classifier

The "Machine Classification View" shown in Figure 3.7 displays results of classification output from the generated classifier. The display consists of both the manual classification (where available) and the automated classification once a classifier has been generated.

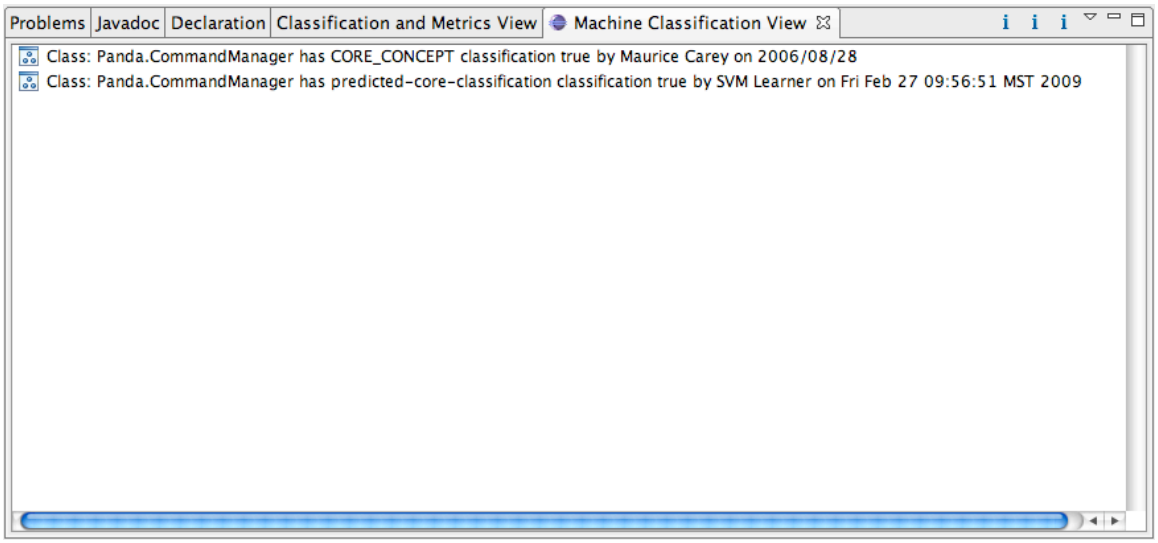

Figure 3.7: Interface for showing classification output

In order to evaluate the accuracy of the classifier, we perform a statistical analysis of the classification results. We calculate the *sample error* from the test set using Equation (3.1) [41] where n is the number of samples in the test set S, f is the function specified by our manual classification mapping the data point x to one of the two classes, h is the *hypothesis* generated by the learning algorithm, and the quantity $\delta(f(x),h(x))$ is defined in Equation (3.2). When the hypothesis disagrees with the manual classification, the sample error increases and $\delta(f(x),h(x))$ is 1 for any instance x where the predicted classification $h(x)$ does not match the expected classification $f(x)$. This allows us to calculate the accuracy of predictions made over a single test set by computing the *sample accuracy* as in Equation (3.3).

$$error_S(h) \equiv \frac{1}{n}\sum_{x\in S}\delta(f(x),h(x)) \tag{3.1}$$

$$\delta(f(x),h(x)) = \begin{cases} 1, & \text{if } f(x) \neq h(x) \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

$$accuracy_S(h) = 1 - error_S(h) \tag{3.3}$$

We can estimate the *true error*, the error of the whole population, from the sample error using Equation (3.4) if we can meet the criteria in Figure 3.8. Here z_N is chosen based on the confidence level of the estimate.

$$error_{\mathscr{D}}(h) = error_S(h) \pm z_N\sqrt{\frac{error_S(h)(1-error_S(h))}{n}} \tag{3.4}$$

Criteria 1 through 4 of Figure 3.8 are met by the methods of selecting the data set and the size of the data set. Criterion 5 is more interesting because the dependency on the probability distribution requires that we test for a significant difference in the distributions of the test samples and the distribution of the population.

1. $n \geq 30$,
2. the hypothesis commits r errors over the n samples,
3. the sample, or test set, S contains n data points which are drawn independently of each other,
4. the sample is chosen independently of the hypothesis, and
5. the sample is chosen according to the probability distribution $\mathscr{D}$ of the population.

Figure 3.8: Criteria for calculation of *true error*

In order to confirm Criterion 5 we can perform the Kolmogorov-Smirnov hypothesis test on the population and sample distributions to test for significant difference. We formulate our hypothesis for this test in Hypothesis 1.

Hypothesis 1. *There is no significant difference in the distribution of the population and the sample data.*

The p-value of the Kolmogorov-Smirnov hypothesis test is the statistical significance, and α is the probability of rejecting the null hypothesis when it is in fact true. We reject the null hypothesis if the p-value is less than α. Rejection of the hypothesis implies that there is a significant difference in the distribution of the sample and the distribution of the population, which then implies that Criterion 5 is not met.

Having shown how we can calculate and verify the *true error*, we really do not need to concern ourselves with the estimate of *true error*. We can easily calculate an average accuracy of hypothesis generated by the learner by running multiple iterations of the same experiment each with randomly sampled training sets. By doing this we eliminate the need to verify Hypothesis 1 for each experiment. Further, we are far more interested in the average accuracy of a hypothesis generated from a randomly selected training set than in the estimated accuracy of some specific hypothesis over some randomly selected test set. This is because of the nature of our approach. We classify some subset of a system, train on that subset, and generate a classifier or a hypothesis as it is referred to in the equations above. So, really we are interested in a learning algorithm that on average generates a hypothesis that is accurate. If we were trying to generate a single hypothesis that was accurate over more than one system, we would be more interested in the *true error*. This

is the method we use in the experiments that we run to establish accuracy of the approach as shown later in Section 3.5.

Using a *t*-test we have a toolset in place to detect situations where, for example, the accuracy is good, but there is no significant difference between the expected value (or mean) of the sample distribution and the measured accuracy. This could happen, as an example, in the case where the distribution of the expected results is 80% negative. The accuracy of the observed results could be 80% just by always guessing negative. The *t*-test can help to discover these situations by testing Hypothesis 2.

Hypothesis 2. *There is no significant difference in the mean of the sample data and the accuracy measured on that sample data.*

Rejecting Hypothesis 2 is a strong indicator that the accuracy is significantly different from the distribution of the data. Intuitively, a significant difference in the accuracy and the mean indicates that the classifier has learned something other than the expected value of the data set.

Threats to Validity

The threats to validity include internal and external factors. The internal factors include errors in the statistical analysis, and overfitting of the data set. The external factors include manual classification errors, and inaccurate or poor definition of the concepts in the domain. We describe each in further detail below.

The primary internal threat to validity that should be addressed is the difficulty in deciding what constitutes a *good result*. The difficulty stems from the distribution of classifications where there are more negatives than positives. In other words there are fewer classes representing concepts than implementation. It has been shown in Section 3.4 that the accuracy of the prediction can not be used in isolation, but must be considered along with the *t*-test analysis of the distributions' expected value versus the accuracy. This gives a more complete picture of the result showing some indication as to how significant the accuracy is. The assumption here was that results that had better than average accuracy along with a significant difference in mean were good results that support our hypothesis.

Overfitting is a threat to validity that effects any machine learning approach, but is only a secondary threat to the validity of our approach. Overfitting means that results need not generalize well to other data sets, because the machine has learned a pattern for a specific data set. In other words if the SVM is overfitted to the data then we could not expect good results from a different data set classified with the same machine. At this point we are not that concerned about generalization to other data sets as we have a practical application of the approach in the classification of a single large system. In future work the potential of overfitting may play a larger role in the threat to internal validity of our approach if our approach is expected to produce general results.

The primary external threat to validity is misclassification during the manual classification process. There are two possible scenarios that result in the introduction of inaccurate analysis into the system and they are either a logical error or a typographic error on the part of the engineer. We have worked with enough software engineers to know that each has a different opinion on any given subject, and these differing opinions would appear in the system as logical errors. However, this assumes that one of the engineers is "more correct" than the other within some externally specified system of objective truth that likely does not exist, and this ignores the possibility of having more than one correct representation of the system. We must concede that any accurate automated classification is only as accurate as the engineer who trained it, but this really is not problematic as long as the engineer is consistent and in fact is representative of the results of a manual classification. The typographic errors that could have been introduced into our data set were minimized by careful data entry, along with rechecking each result against the class. Though we believe we have minimized the errors as much as possible we have no way to measure this using the current process, and as is the case with many machine learning applications we can not predict the effect of a classification error on the resulting classifier.

A secondary external threat to validity is the design of the software itself. If poorly designed software systems are introduced it is difficult to predict the results. As an example, imagine a single concept being represented by two classes. There may be instances where the lack of cohesion makes sense but it may not be possible to decide where.

A third external threat to validity is the selection of the software itself. It might be argued that the systems we classify here are not representative of software in general, or it may be that these systems are special in some way that our technique works well enough here but not on other systems. It is true that the systems we have selected may not represent the use of software in general, and they are special in the sense that we developed criteria for there selection but this did not include any sorting based on performance. This is unavoidable as we are looking only at systems written in Java, that have open source, and that are currently maintained. Our goal is to select a few systems that have different functionality within that space of currently maintained open source Java software projects. There are at least two ways to approach this issue. We could show that the systems we have selected are a reasonably representative sample of the population of all software. This is logistically impossible given the lack of access to commercial software, or the metrics from that software. Even given the data and time to process it the scope of the work to determine in what dimension the measurement should take place is well beyond our current knowledge of these systems. For example, we might say that we measure all classes in the given population using the metrics we have defined here then go on to show that the classes in the systems we have experimented with are representative of those from the population, but then the question still remains: Is there something special about the relationships of those metrics in the systems we have chosen? We believe that regardless of how we measure we would arrive at the same issue of these measures being unsuitable for determining applicability of our approach to any given system because statistical analysis simply is not the correct tool to mitigate this threat. Ultimately it is only with a deeper understanding of the underlying reason for the success of our approach that a formulation of a solution to this threat can be found.

3.5 Preliminary Results

In our initial investigations we have shown that statistically significant accuracy can be achieved. We form Hypothesis 2 based on the first question in our first goal. We can reject this hypothesis if a *t*-test shows a significant difference in the average accuracy of an experiment verses the median of the data set. Rejection of this hypothesis implies that statistically significant accuracy can be a achieved and we have an affirmative answer to our question.

Experiments

This section shows the results of the ten experiments we have conducted using the approach outlined in Section 3.4. We do not count experiments 1 and 7 as they are designed to collect the baseline accuracy given a simple learner that finds the median value of the classification outcome. The results are summarized in Table 3.1, which shows the average accuracy, the size of the training set, and the total size of, or number of classes in, the system.

Table 3.1: Summary of Average Accuracy for Preliminary Results

Experiment	Average Accuracy	Training Set Size	Classes in System
1 Panda Default	69.39%	35	90
2 Panda KNN	73.54%	35	90
3 Panda SVM	75.65%	35	90
4 Panda ID3	81.09%	35	90
5 Panda ID3 + Ada Boost	78.91%	35	90
6 Panda Vote	73.54%	35	90
7 Scarab Default	69.74%	200	580
8 Scarab KNN	75.18%	200	580
9 Scarab SVM	76.25%	200	580
10 Scarab ID3	81.81%	200	580
11 Scarab ID3 + Ada Boost	83.25%	200	580
12 Scarab Vote	81.18%	200	580

Table 3.2: Results of Hypothesis 2 *t*-test.

Experiment	p-value	Reject Hypothesis 2
Panda KNN	< 0.001	Yes
Panda SVM	< 0.001	Yes
Panda ID3	< 0.001	Yes
Panda ID3 + Ada Boost	< 0.001	Yes
Panda Vote	< 0.001	Yes
Scarab KNN	< 0.001	Yes
Scarab SVM	< 0.001	Yes
Scarab ID3	< 0.001	Yes
Scarab ID3 + Ada Boost	< 0.001	Yes
Scarab Vote	< 0.001	Yes

Experiment 1: Panda Default Learner

For this experiment we used the data set collected from Panda using the methods outlined in Section 3.2 of Chapter 3. The process of classifying Panda took approximately four hours but given our previous experience with the software this may not be an indicator of the time required for an engineer looking at the system for the first time. We followed the process outlined in Section 3.4 of Chapter 3 to split the data into training and test sets, and used the training set to generate the default classifier used in this experiment. The experiment was repeated 30 times and the average accuracy was calculated as the mean of the accuracies for each run. The purpose of this experiment is to establish a baseline value for accuracy by learning the median value of the system.

Results are shown in Table 3.1. Accuracy was 69.39% for this sample, which represents the the expected value of the data set used in the *t*-test analysis shown in Table 3.2.

Experiment 2: Panda KNN

For this experiment we used the data set collected from Panda using the methods outlined in Section 3.2 of Chapter 3. We followed the process outlined in Section 3.4 of Chapter 3 to split the data into training and test sets, and used the training set to generate the KNN classifier used in this experiment.

We used a KNN classifier with a K value of 3 and distance calculation based on simple Euclidean distance, meaning the classification is based on a majority rules vote of the three closest training set data points in *n* dimensional space. Results are shown in Table 3.1. Average accuracy was 73.54% for this sample with 30 runs of the experiment. The *t*-test analysis (as indicated in Table 3.2) showed that the accuracy of the test versus the expected value of the results collected from the software were significantly different at the 99.9% confidence level, in other words the probability that the accuracy is different than the mean of the distribution is over 99.9%. Precision and recall for this experiment are shown in Table 3.3.

Experiment 3: Panda SVM

For this experiment we used the data set collected from Panda using the methods outlined in Section 3.2 of Chapter 3. We followed the process outlined in Section 3.4 of Chapter 3 to split the

Table 3.3: Precision and Recall for Panda KNN

	actual true	actual false	class precision
predicted true	244	169	59.08%
predicted false	146	701	82.76%
class recall	62.56%	80.57%	

Table 3.4: Precision and Recall for Panda SVM

	actual true	actual false	class precision
predicted true	131	8	94.24%
predicted false	259	862	76.90%
class recall	33.59%	99.08%	

data into training and test sets, and used the training set to generate the SVM classifier used in this experiment. The purpose of this experiment was to attempt to identify evidence for the hypothesis using a fairly small system. We wanted to see if it was worth while to commit to the process for a larger system. The data set for Panda is under 100 elements and while a larger data set would better capture any generalizable properties of the hypothesis, this data set was simple to obtain and encouraged further analysis.

We used a SVM classifier based on a radial basis function kernel as in Equation (2.6) of Section 2.4 with the slack term of $C = 0$. Results are shown in Table 3.1. Accuracy was 77.55% for this sample. The *t*-test analysis (as indicated in Table 3.2) showed that the accuracy of the test versus the expected value of the results collected from the software were significantly different at the 99.9% confidence level. These results were encouraging for a data set of this size, prompting us to proceed with the data collection effort for Scarab. Precision and recall for this experiment are shown in Table 3.4.

Experiment 4: Panda ID3

For this experiment we used the data set collected from Panda using the methods outlined in Section 3.2 of Chapter 3. We followed the process outlined in Section 3.4 of Chapter 3 to split the data into training and test sets, and used the training set to generate the ID3 classifier used in this experiment.

Table 3.5: Precision and Recall for Panda ID3

	actual true	actual false	class precision
predicted true	287	119	70.69%
predicted false	103	751	87.94%
class recall	73.59%	86.32%	

Table 3.6: Precision and Recall for Panda ID3 with Aba Boost

	actual true	actual false	class precision
predicted true	319	289	52.47%
predicted false	71	581	89.11%
class recall	81.79%	66.78%	

We used an ID3 classifier with a minimal leaf size of 2 and minimal gain of 0.02. Results are shown in Table 3.1. Average accuracy was 81.09% for this sample with 30 runs of the experiment. The *t*-test analysis (as indicated in Table 3.2) showed that the accuracy of the test versus the expected value of the results collected from the software were significantly different at the 99.9% confidence level. Precision and recall for this experiment are shown in Table 3.5.

Experiment 5: Panda ID3 + Ada Boost

For this experiment we used the data set collected from Panda using the methods outlined in Section 3.2 of Chapter 3. We followed the process outlined in Section 3.4 of Chapter 3 to split the data into training and test sets, and used the training set to generate the ID3 decision tree with ada boosting classifier used in this experiment.

We used an ID3 classifier with a minimal leaf size of 2 and minimal gain of 0.02. The Ada Boost was configured for 10 iterations. Results are shown in Table 3.1. Average accuracy was 78.91% for this sample with 30 runs of the experiment. The *t*-test analysis (as indicated in Table 3.2) showed that the accuracy of the test versus the expected value of the results collected from the software were significantly different at the 99.9% confidence level. Precision and recall for this experiment are shown in Table 3.6.

Table 3.7: Precision and Recall for Panda Vote

	actual true	actual false	class precision
predicted true	227	34	86.97%
predicted false	163	836	83.68%
class recall	58.21%	96.09%	

Experiment 6: Panda Vote

For this experiment we used the data set collected from Panda using the methods outlined in Section 3.2 of Chapter 3. We followed the process outlined in Section 3.4 of Chapter 3 to split the data into training and test sets, and used the training set to generate the SVM, ID3, and KNN classifiers used in this experiment.

We used all classifiers as configured in previous experiments but with a simple majority rules vote in order to arrive at the classification value. Results are shown in Table 3.1. Average accuracy was 73.54% for this sample with 30 runs of the experiment. The *t*-test analysis (as indicated in Table 3.2) showed that the accuracy of the test versus the expected value of the results collected from the software were significantly different at the 99.9% confidence level. Precision and recall for this experiment are shown in Table 3.7.

Experiment 7: Scarab Default Learner

For this experiment we used the data set collected from Scarab using the methods outlined in Section 3.2 of Chapter 3. Classification of Scarab required significantly more time than classifying Panda with over 40 hours of effort expended. We followed the process outlined in Section 3.4 of Chapter 3 to split the data into training and test sets, and used the training set to generate the default classifier used in this experiment. The experiment was repeated 30 times and the average accuracy was calculated as the mean of the accuracies for each run. The purpose of this experiment is to establish a baseline value for accuracy by learning the median value of the system.

Results are shown in Table 3.1. Accuracy was 69.74% for this sample, which represents the the expected value of the data set used in the *t*-test analysis shown in Table 3.2.

Table 3.8: Precision and Recall for Scarab KNN

	actual true	actual false	class precision
predicted true	1299	659	66.34%
predicted false	1341	5401	80.11%
class recall	49.20%	89.13%	

Experiment 8: Scarab KNN

For this experiment we used the data set collected from Scarab using the methods outlined in Section 3.2 of Chapter 3. We followed the process outlined in Section 3.4 of Chapter 3 to split the data into training and test sets, and used the training set to generate the KNN classifier used in this experiment.

In this experiment we used a value of $k = 3$. Results are shown in Table 3.1. The only parameter to the k-nearest neighbor algorithm is k. Average accuracy was 75.18% for this sample with 30 runs of the experiment. The t-test analysis (as indicated in Table 3.2) showed that the accuracy of the test versus the expected value of the results collected from the software were significantly different at the 99.9% confidence level. Precision and recall for this experiment are shown in Table 3.8.

Experiment 9: Scarab SVM

For this experiment we used the data set collected from Scarab. Classification of Scarab required significantly more time than classifying Panda with over 40 hours of effort expended. This may have been a function of less familiarity with the system as well as the larger size. The process outlined in Section 3.4 was used to split the data into training and test sets, generate an SVM classifier for this experiment, and collect results. The purpose of this experiment was to validate the approach for a much larger data set than that used in the Panda experiment. Scarab is also a program that is more representative of systems in actual use.

The SVM classifier used was based on a radial basis function kernel with parameter of $C = 0$. Results are shown in Table 3.1. Average accuracy was 72.49% for this sample with 30 runs of the experiment. The t-test analysis (as indicated in Table 3.2) showed that the accuracy of the test

Table 3.9: Precision and Recall for Scarab SVM

	actual true	actual false	class precision
predicted true	618	257	70.63%
predicted false	2022	5803	74.16%
class recall	23.41%	95.76%	

Table 3.10: Precision and Recall for Scarab ID3

	actual true	actual false	class precision
predicted true	1926	634	75.23%
predicted false	714	5426	88.37%
class recall	72.95%	89.54%	

versus the expected value of the results collected from the software were significantly different at the 99.9% confidence level. Precision and recall for this experiment are shown in Table 3.9.

Experiment 10: Scarab ID3

For this experiment we used the data set collected from Scarab using the methods outlined in Section 3.2 of Chapter 3. We followed the process outlined in Section 3.4 of Chapter 3 to split the data into training and test sets, and used the training set to generate the ID3 classifier used in this experiment.

We used an ID3 classifier with a minimal leaf size of 2 and minimal gain of 0.02. Results are shown in Table 3.1. Average accuracy was 81.81% for this sample with 30 runs of the experiment. The *t*-test analysis (as indicated in Table 3.2) showed that the accuracy of the test versus the expected value of the results collected from the software were significantly different at the 99.9% confidence level. Precision and recall for this experiment are shown in Table 3.10.

Experiment 11: Scarab ID3 + Ada Boost

For this experiment we used the data set collected from Scarab using the methods outlined in Section 3.2 of Chapter 3. We followed the process outlined in Section 3.4 of Chapter 3 to split the data into training and test sets, and used the training set to generate the ID3 decision tree with ada boosting classifier used in this experiment.

Table 3.11: Precision and Recall for Scarab ID3 with Ada Boost

	actual true	actual false	class precision
predicted true	2400	1413	62.94%
predicted false	240	4647	95.09%
class recall	90.91%	76.68%	

We used an ID3 classifier with a minimal leaf size of 2 and minimal gain of 0.02. The Ada Boost was configured for 10 iterations. Results are shown in Table 3.1. Average accuracy was 83.25% for this sample with 30 runs of the experiment. The *t*-test analysis (as indicated in Table 3.2) showed that the accuracy of the test versus the expected value of the results collected from the software were significantly different at the 99.9% confidence level. This experiment has the highest average accuracy of any experiment we have performed. This seems to indicate that ID3 with boosting may be a good candidate for further evaluation. Precision and recall for this experiment are shown in Table 3.11.

Experiment 12: Scarab Vote

For this experiment we used the data set collected from Scarab using the methods outlined in Section 3.2 of Chapter 3. We followed the process outlined in Section 3.4 of Chapter 3 to split the data into training and test sets, and used the training set to generate the SVM, ID3, and KNN classifiers used in this experiment.

We used all classifiers as configured in previous experiments but with a simple majority rules vote in order to arrive at the classification value. Results are shown in Table 3.1. Average accuracy was 81.18% for this sample with 30 runs of the experiment. The *t*-test analysis (as indicated in Table 3.2) showed that the accuracy of the test versus the expected value of the results collected from the software were significantly different at the 99.9% confidence level. Precision and recall for this experiment are shown in Table 3.12.

Discussion of Results.

Looking at the precision and recall we see some patterns emerging. It appears that ID3 is able to achieve respectable precision and recall results considering we have yet to filter the feature set. It also seems that applying Ada Boost to ID3 helps to reduce the number of false negatives.

Table 3.12: Precision and Recall for Scarab Vote

	actual true	actual false	class precision
predicted true	1180	383	75.50%
predicted false	1460	5677	79.54%
class recall	44.70%	93.68%	

Table 3.2 gives us an indicator of how well the accuracy is measured. A small p-value is an indicator that there is a significant difference between the accuracy and the mean. Imagine a coin that is unfairly biased towards heads such that the expected value of heads is 80%. If we were to play a game where the coin is flipped repeatedly and the player must try to guess the result, the player would eventually discover the bias and begin guessing heads every round of the game. This learned strategy would lead to a measured accuracy of the player at about 80%, in other words there would be no significant difference between the expected value of the experiment and the accuracy of the guess. In our results we believe that the machine is really making good predictions, because there *is* a statistically measurable difference between the expected value of the data set and the classification guess of the machine for all of the classifiers' predictions. So the results show support for rejecting Hypothesis 2. By rejecting Hypothesis 2 we find support for our informal hypothesis that there exist a set of metrics that capture properties of a system which can be used by a machine to classify concepts of that system. In the following chapter we determine if it is practical to use this approach.

Chapter 4

Improvements, Practicality, and Validation of the Approach

In this chapter we address improvements that can be made to the approach, the practicality of using such an approach in the real world, and validation of the approach. We address improvements and practicality via feature set minimization and training set efficiency. We validate via a study of a third party applying the approach independently to one of the software systems we have studied.

4.1 Minimizing the Feature Set

The primary motivations for minimizing the feature set is to increase understanding of what the machine learning algorithms are learning, to simplify the computations involved in the process, and to potentially increase the accuracy of the classification process.

Feature set selection is used to discover the optimal set of metrics which provide the highest level of classification accuracy for a problem. While feature set selection is a computationally complex problem, a solution can be approximated via one of several algorithms including Sequential Forward Selection [56] also known as forward selection, Sequential Backward Selection [37] also known as backward elimination, and Evolutionary Selection [51].

The sequential forward selection algorithm works by first selecting the single feature that has the greatest predictive accuracy of the classification. Next, other features are added one at a time; the one with the greatest increase is made part of the selection. This process continues until no further improvements are made. However, additional iterations can be made to ensure that local optima do not fool the algorithm. In our use of forward selection, we use a minimum 5% increase in relative performance as a stopping criterion and allow 6 speculative iterations.

The backward elimination algorithm works by starting with the full feature set and eliminating one feature in each iteration. Each iteration consists of a cross validation on the input data for each remaining feature resulting in a performance metric. The feature with the lowest decrease in performance is then eliminated. This process continues until no further features can be eliminated

without a specified decrease in performance. Similar to forward selection there is also a specified number of additional iterations that can help to reduce local optima. In our use of backward elimination, we selected a 10% decrease in relative performance as a stopping criterion and allow 6 speculative iterations.

The evolutionary selection algorithm generates a population of individuals each with a feature switched on with a given probability. Once initialized, the algorithm mutates each individual in the population by setting used features to unused and vice versa with a given mutation probability resulting in an increase to the population. After the mutation step, the algorithm randomly selects two individuals from the population and performs crossover with the given crossover probability. Finally, each individual in the population is conceptually mapped to a section of a roulette wheel proportional to the individual's fitness and the given population size. Then individuals are drawn at random. As long as fitness improves from one generation to the next we loop back to the mutation step, unless we exceed a maximum number of generations. In our experiments, we use a population size of 5, a feature initialization probability of 0.5, a mutation probability of $1/n$ where n is the number of attributes, a crossover probability of 0.5, and a maximum number of generations of 60.

Results of Feature Set Selection

To determine the set of features with the highest contribution to performance in this learning problem we perform several rounds of feature selection. We use multiple rounds, because the feature selection algorithms are heuristics. By eliminating the features that contribute the least to performance after each round we are able to find a better heuristic solution in the next round. Repeating the accuracy experiments described in Section 3.5 may help to verify that this assumption is true; however it is possible given the heuristic nature of these algorithms that an important feature is eliminated. Given the aggressive nature of our evolutionary algorithm along with validation in Section 4.1 this appears to be an exceedingly rare possibility.

We begin our analysis of the first round of selection by looking at the metrics VG, and WMC in Table 4.1 on page 47, Table 4.2 on page 48, and Table 4.3 on page 49. VG was not found to contribute to increased accuracy by any of the selection algorithms used with any of the machine

learning methods. Therefore we eliminate VG from consideration. VG is related to WMC as it is calculated as WMC averaged over the number of methods in the class. WMC is selected only by the more aggressive evolutionary selection algorithm for use with ID3, so is also eliminated in the first round of selection.

Table 4.1: First Round of Forward Selection

Metric	KNN	SVM	ID3
DIT	1	1	1
LCOM	0	1	0
LOC	0	0	0
VG	0	0	0
MLOC	0	0	0
NBD	0	0	0
NOF	0	0	0
NOM	1	0	0
NORM	0	0	1
PARC	0	0	0
NSF	0	0	1
NSM	0	0	0
NSC	0	0	0
SIX	0	1	0
USEDBY	1	1	1
USES	0	0	0
WMC	0	0	0

Next we look at several other metrics besides WMC that were only selected in one instance of the feature selection runs. These include NBD, NOF, NORM, NSF, and NSC. There is little evidence to suggest any of the metrics selected in individual experimental instances have significant contribution to performance. Therefore we eliminate these metrics from consideration as well.

We begin the second round of selection with the ten remaining metrics listed in Table 4.4 on page 49, Table 4.5 on page 50, and Table 4.6 on page 50. We repeat the experiments in the first round but the metrics previously eliminated have been filtered from the input. For this round we eliminate any metric that is not selected by at least one experimental instance. Metrics not found to contribute to accuracy in this round are LOC, NOM, and NSM.

Table 4.2: First Round of Backward Elimination

Metric	KNN	SVM	ID3
DIT	0	1	1
LCOM	0	0	0
LOC	1	0	0
VG	0	0	0
MLOC	0	0	0
NBD	0	0	0
NOF	0	0	0
NOM	0	0	0
NORM	0	0	0
PARC	0	0	0
NSF	0	0	0
NSM	0	0	0
NSC	0	0	0
SIX	0	0	0
USEDBY	1	1	1
USES	0	0	0
WMC	0	0	0

We begin the third round of selection with the seven metrics listed in Table 4.7 on page 51. For this round we only perform the evolutionary selection based experiments as the previous forward selection and backward elimination showed little difference from round one to round two. From the results we see DIT, LCOM, PARC, and USEDBY are selected for all three learning algorithms.

Validation of Feature Set Selection

In order to validate our feature selection results we repeat the experiments performed in Section 3.5 with the input metrics filtered to remove those metrics eliminated by our feature selection criteria. The expectation is that our accuracy results should either stay the same or improve over previous results.

We can see in Table 4.8 that the accuracy has improved significantly over the results prior to feature selection. The repeated experiments are marked with FS to signify the input metrics have been filtered to include only those metrics identified by our feature selection process. No other aspect of the experiment was changed. Default experiments are not repeated for the filtered

Table 4.3: First Round of Evolutionary Selection

Metric	KNN	SVM	ID3
DIT	1	1	1
LCOM	1	0	0
LOC	0	1	1
VG	0	0	0
MLOC	0	1	1
NBD	1	0	0
NOF	0	1	0
NOM	0	1	0
NORM	0	0	0
PARC	1	1	1
NSF	0	0	0
NSM	1	1	1
NSC	1	0	0
SIX	0	0	1
USEDBY	1	1	1
USES	1	1	0
WMC	0	0	1

Table 4.4: Second Round of Forward Selection

Metric	KNN	SVM	ID3
DIT	1	1	1
LCOM	0	0	0
LOC	0	0	0
MLOC	0	0	0
NOM	0	0	0
PARC	0	0	0
NSM	0	0	0
SIX	0	0	0
USEDBY	1	1	1
USES	0	0	0

features as these experiments do not depend on the input metrics and produce the same results. They are included here only for comparison.

In Table 4.9 the difference in precision and recall are shown between the initial experiments as discussed in Section 3.5 and the experiments performed here after filtering based on feature selection. Complete confusion matrices for each post feature selection experiment performed are

Table 4.5: Second Round of Backward Elimination

Metric	KNN	SVM	ID3
DIT	1	0	0
LCOM	0	0	0
LOC	0	0	0
MLOC	0	0	0
NOM	0	0	0
PARC	0	0	0
NSM	0	0	0
SIX	0	0	0
USEDBY	1	1	1
USES	0	0	0

Table 4.6: Second Round of Evolutionary Selection

Metric	KNN	SVM	ID3
DIT	1	1	1
LCOM	1	1	0
LOC	0	0	0
MLOC	0	1	1
NOM	0	0	0
PARC	1	1	0
NSM	0	0	0
SIX	0	0	1
USEDBY	1	1	1
USES	0	1	0

found in Appendix A. There are a few interesting trends. First, SVM increased in recall of concepts significantly for both Panda and Scarab. Second, ID3 with Ada Boost has better recall of concepts prior to feature selection. Third, KNN also increase in recall of concepts for both Panda and Scarab. Finally, ID3 is nearly identical in concept recall performance after feature selection, most of the performance increase seen in Table 4.8 was based on increased non-concept recall.

4.2 Training Set Efficiency and Size

We examine training set efficiency and size in order to determine if adequate accuracy can be achieved with a reasonably sized training set. The terms adequate and reasonable are obviously vague. An issue exists in defining adequate accuracy and a reasonably sized training set since these

Table 4.7: Third Round of Evolutionary Selection

Metric	KNN	SVM	ID3
DIT	1	1	1
LCOM	1	1	1
MLOC	0	0	1
PARC	1	1	1
SIX	1	0	0
USEDBY	1	1	1
USES	0	1	0

are both very dependent on the needs and opinions of the user of the approach. For many, manually classifying 10% of a large system with the expectation of producing a classifier that can predict concepts with 80 to 90% accuracy is reasonable. For others it may not be. This largely depends on why concepts are being discovered.

In this section rather than trying to determine the needs of an individual user of this technique we instead describe the accuracy that can be expected using this approach for varying sizes of training sets. This in turn enables the user of this approach to determine if it is adequate for their needs. We also examine different techniques for picking training sets. We compare the accuracy obtained for each of three different selection techniques on varying sized training sets. We treat different learning algorithms in hopes of identifying any failures or advantages of particular algorithms.

Learning Curve Results

A learning curve shows the relationship between predictive performance and learning effort as measured by the size of the training set [15]. By examining the learning curves we can determine what training set size is needed to obtain a given performance level, what algorithms are best suited to the problem, and which method of training set selection is most effective. We present the learning curves here grouped in each of the figures by dataset and learning algorithm. We use diamonds to represent points on the learning curve for the random training set selection method, we use circles for the Kennard-Stone method, and triangles for the reverse Kennard-Stone method.

Figure 4.1 shows the learning curves for the KNN algorithm executed on the Panda dataset for each of the training set selection methods. We can see the challenges of a very small dataset in this

Table 4.8: Summary of Average Accuracy for Preliminary versus Feature Selection Results

Experiment	Average Accuracy	Training Set Size	Classes in System
1 Panda Default	69.39%	35	90
2 Panda KNN	73.54%	35	90
2-FS Panda KNN	88.41%	35	90
3 Panda SVM	75.65%	35	90
3-FS Panda SVM	89.29%	35	90
4 Panda ID3	81.09%	35	90
4-FS Panda ID3	84.21%	35	90
5 Panda ID3 + Ada Boost	78.91%	35	90
5-FS Panda ID3 + Ada Boost	82.86%	35	90
6 Panda Vote	73.54%	35	90
6-FS Panda Vote	87.75%	35	90
7 Scarab Default	69.74%	200	580
8 Scarab KNN	75.18%	200	580
8-FS Scarab KNN	88.37%	200	580
9 Scarab SVM	76.25%	200	580
9-FS Scarab SVM	86.29%	200	580
10 Scarab ID3	81.81%	200	580
10-FS Scarab ID3	84.52%	200	580
11 Scarab ID3 + Ada Boost	83.25%	200	580
11-FS Scarab ID3 + Ada Boost	84.40%	200	580
12 Scarab Vote	81.18%	200	580
12-FS Scarab Vote	87.75%	200	580

figure. With 90 data points in the complete system the first four data points in the figure represent training sets of size 4, 6, 9, and 11 respectively. When we move to the fifth data point for random selection representing a training set of size 13 we see a drop in performance. This is due to the relatively small number of data points in the training set. The probability of a single data point being added to the training set having an adverse effect on performance is higher since the single data point represents a greater portion of the addition to the set. This effect is lessened as the training set grows to a larger portion of the complete system. Looking at Kennard-Stone we see that we never achieve accuracy greater than that expected of a default learner. Reverse Kennard-Stone does eventually achieve accuracy better than that expected of a default learner but it begins with abysmal results, and seems to be even more affected by the small data set size than

Table 4.9: Post Feature Selection Precision and Recall

	Recall		Precision	
Experiment	true Concept	true Non-Concept	predicted Concept	predicted Non-Concept
2 Panda KNN	62.56%	80.57%	59.08%	82.76%
2-FS Panda KNN	80.26%	92.07%	81.94%	91.23%
3 Panda SVM	33.59%	99.08%	94.24%	76.90%
3-FS Panda SVM	71.54%	97.24%	92.08%	88.40%
4 Panda ID3	73.59%	86.32%	70.69%	87.94%
4-FS Panda ID3	73.33%	89.08%	75.07%	88.17%
5 Panda ID3 + Ada Boost	81.79%	66.78%	52.47%	89.11%
5-FS Panda ID3 + Ada Boost	73.85%	86.90%	71.64%	88.11%
6 Panda Vote	58.21%	96.09%	86.97%	83.68%
6-FS Panda Vote	78.21%	94.83%	87.14%	90.66%
8 Scarab KNN	49.20%	89.13%	66.34%	80.11%
8-FS Scarab KNN	80.72%	91.70%	80.90%	91.61%
9 Scarab SVM	23.41%	95.76%	70.63%	74.16%
9-FS Scarab SVM	71.06%	92.92%	81.39%	88.05%
10 Scarab ID3	72.95%	89.54%	75.23%	88.37%
10-FS Scarab ID3	72.23%	89.87%	75.64%	88.14%
11 Scarab ID3 + Ada Boost	90.91%	76.68%	62.94%	95.09%
11-FS Scarab ID3 + Ada Boost	75.76%	88.17%	73.61%	89.30%
12 Scarab Vote	44.70%	93.68%	75.50%	79.54%
12-FS Scarab Vote	76.52%	92.64%	81.91%	90.05%

random selection since there are three points where additional learning results in reduced performance.

Figure 4.2 shows the learning curves for the SVM algorithm executed on the Panda dataset for each of the training set selection methods. The challenges of a small data set are still present in this case. Comparing to Figure 4.1, there are some similarities with the performances of each of the different training set selection types though they appear to reach the higher levels of accuracy only after larger training sets. For example the random selection method eventually reaches the same steady state accuracy level but only after seeing about 17% more of the system.

Figure 4.3 shows the learning curves for the ID3 numerical algorithm executed on the Panda dataset for each of the training set selection methods. The learning curves illustrate an issue with the ID3 numerical algorithm applied to this problem. Notice that the performance tends to vary a great deal from one training set size to another. This effect is created by two properties of the ID3

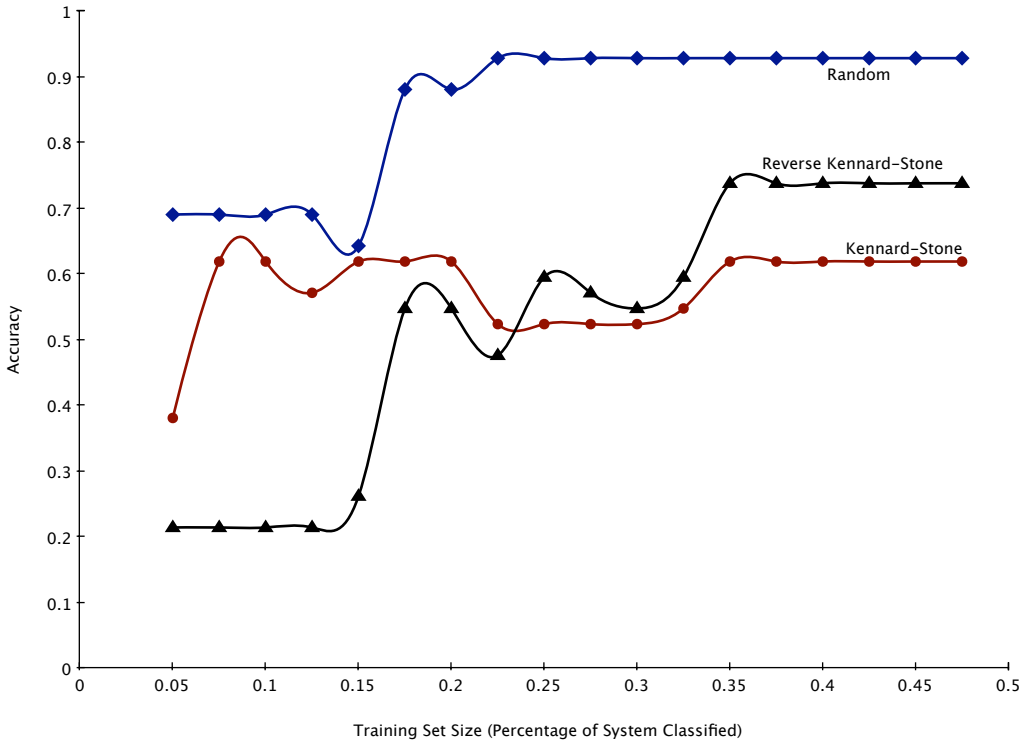

Figure 4.1: Learning Curve for KNN on Panda

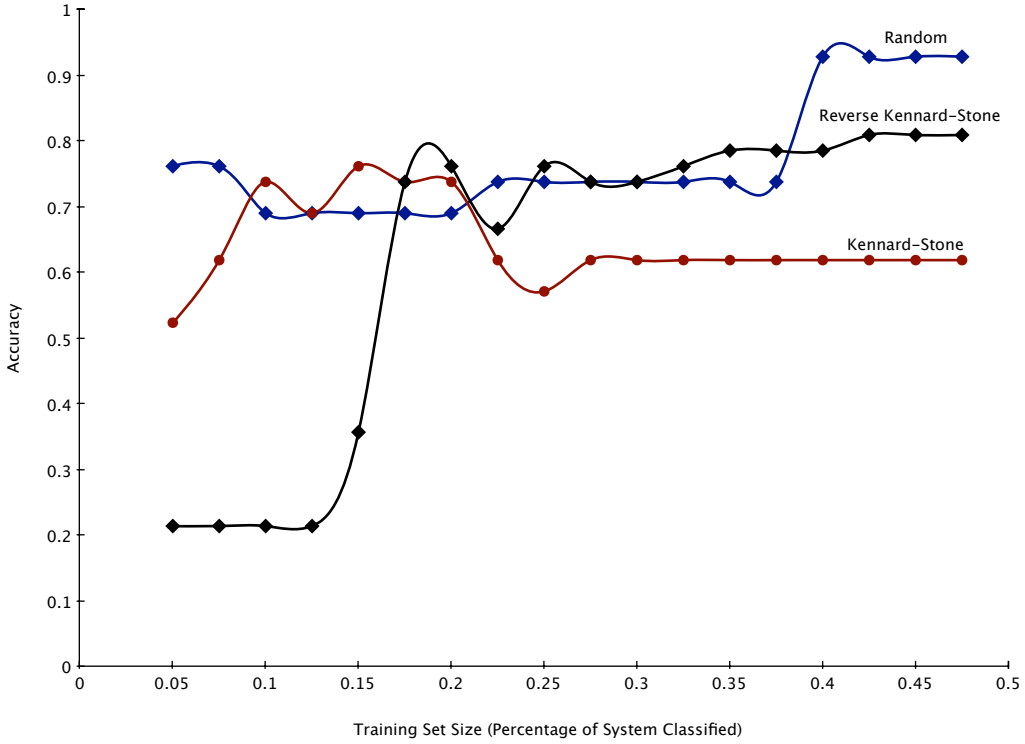

Figure 4.2: Learning Curve for SVM on Panda

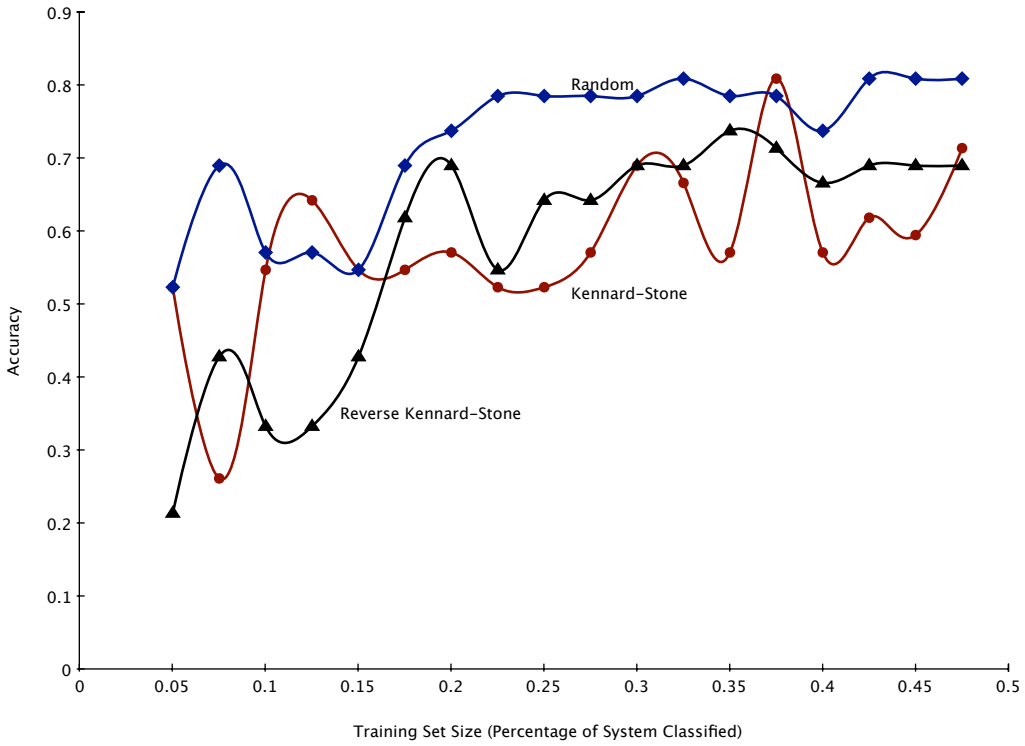

Figure 4.3: Learning Curve for ID3 on Panda

numerical algorithm. First, ID3 is susceptible to over-fitting of training data as it creates a perfect classifier for the training set [47, 41]. Second, the original ID3 algorithm was designed for discrete data and has been adapted to continuous numerical data via use of a binning technique. This binning technique when used on the incomplete data present in the training set produces bins that could be said to have a certain noise level. To illustrate, consider a simple example of one attribute with values 1.1, 1.2, and 1.3 where 1.1 and 1.2 are classified in the A class and 1.3 is in the B class. Our ID3 numerical algorithm chooses some value c that divides the examples into appropriate classes. Let us assume this value is simply a linear interpolation between values on a class boundary, so 1.25 is selected. We now have two leaf nodes in our decision tree, one node says anything with value less than 1.25 is class A, and one says anything greater than 1.25 is class B. Now if we have an example in our test set which is class B but has an attribute value of 1.22 we have a classification error. This error is based on noise in the system introduced by the binning algorithm. It is also interesting that by introducing additional values into the training set the cut point of the classification boundary can move about the test point such that the classifier becomes

more or less accurate for that sample based on the training set. Quinlan [47] examined the consequences of noise on attributes and found that if noise is present on all attributes, as it would be in our problem based on binning of continuous values, the degradation in classification performance can be up to 30%. Given the small size of the Panda data set, we cannot say for certain that this is the cause of the variances in the learning curve, however if we encounter similar variances in the larger Scarab data set we would be inclined to believe that ID3 is not an appropriate algorithm for this problem.

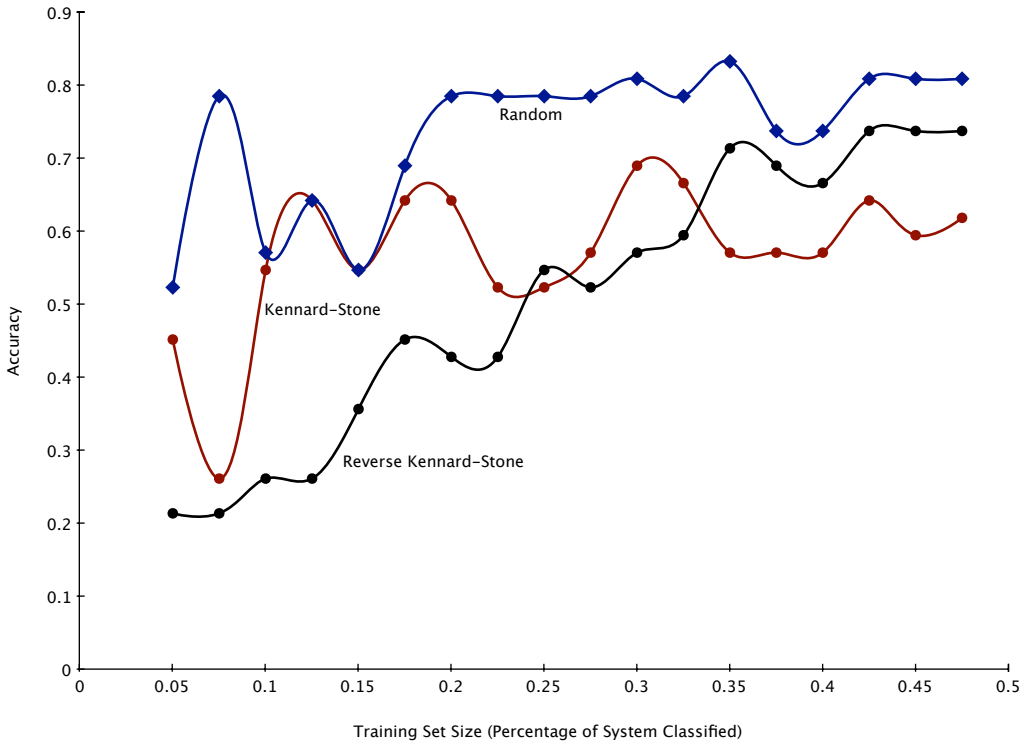

Figure 4.4: Learning Curve for Vote on Panda

Figure 4.4 shows the learning curves for the Vote algorithm executed on the Panda dataset for each of the training set selection methods. This is interesting because basically we are averaging the results of KNN, SVM, and ID3 via a majority rules vote. This is reflected in the learning curves which display an averaging of the properties from the learning curves of each of the individual algorithms. What is clear in this figure is that random selection is the most consistently performing selection method for the Panda data set for the learning algorithms tested.

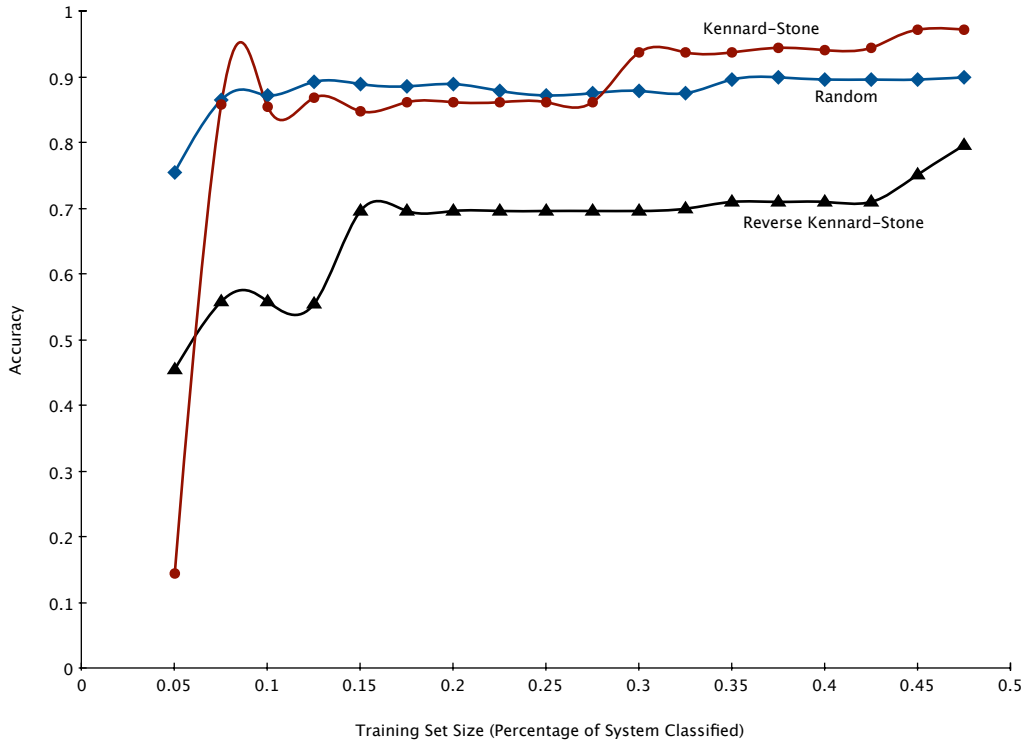

Figure 4.5: Learning Curve for KNN on Scarab

Figure 4.5 shows the learning curves for the KNN algorithm executed on the Scarab dataset for each of the training set selection methods. Our analysis of Panda is obviously limited due to the small data set; however here we see clear patterns emerging. The initial data point with 5% of the system classified, which is a training set with 29 data points, shows very poor results for Kennard-Stone. After the first data point Kennard-Stone nearly matches the random selection method until the training set grows to 174 data points at 30% of the system classified then it surpasses the random selection method.

Figure 4.6 shows the learning curves for the SVM algorithm executed on the Scarab dataset for each of the training set selection methods. As with the Panda training set, there is a similarity to the results for KNN in Figure 4.5. It is interesting that the highest performance levels are below those reached in the KNN experiments. This seems to indicate that the additional power of the SVM algorithm is not being put to good use with this particular problem.

Figure 4.7 shows the learning curves for the ID3 numerical algorithm executed on the Scarab dataset for each of the training set selection methods. We again encounter the swinging variances

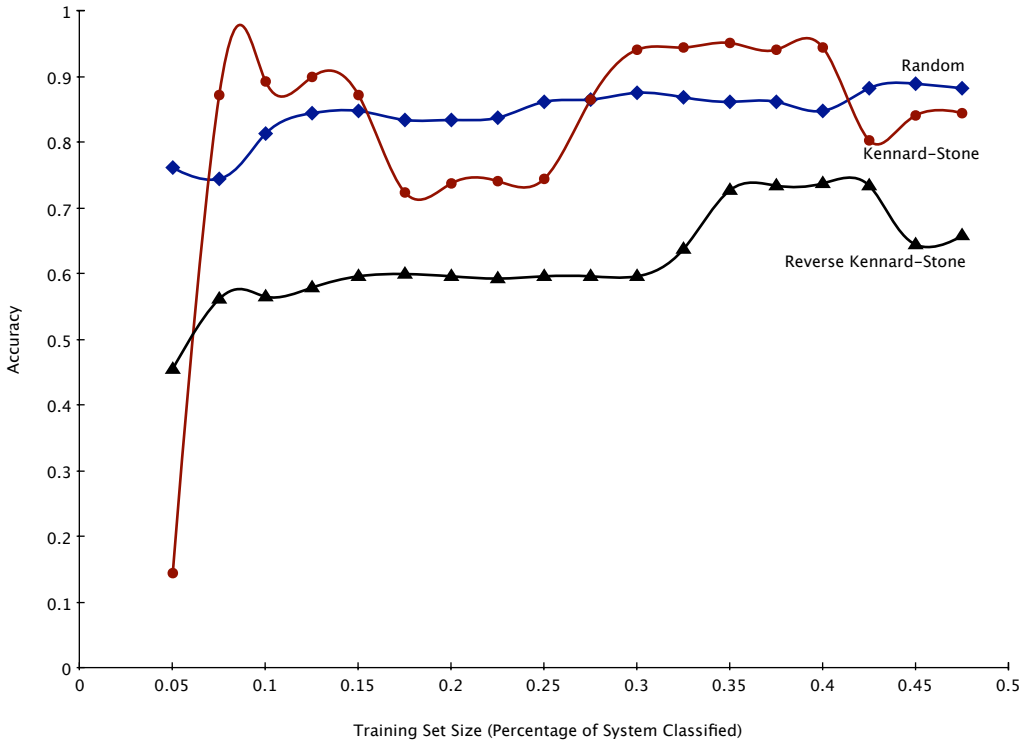

Figure 4.6: Learning Curve for SVM on Scarab

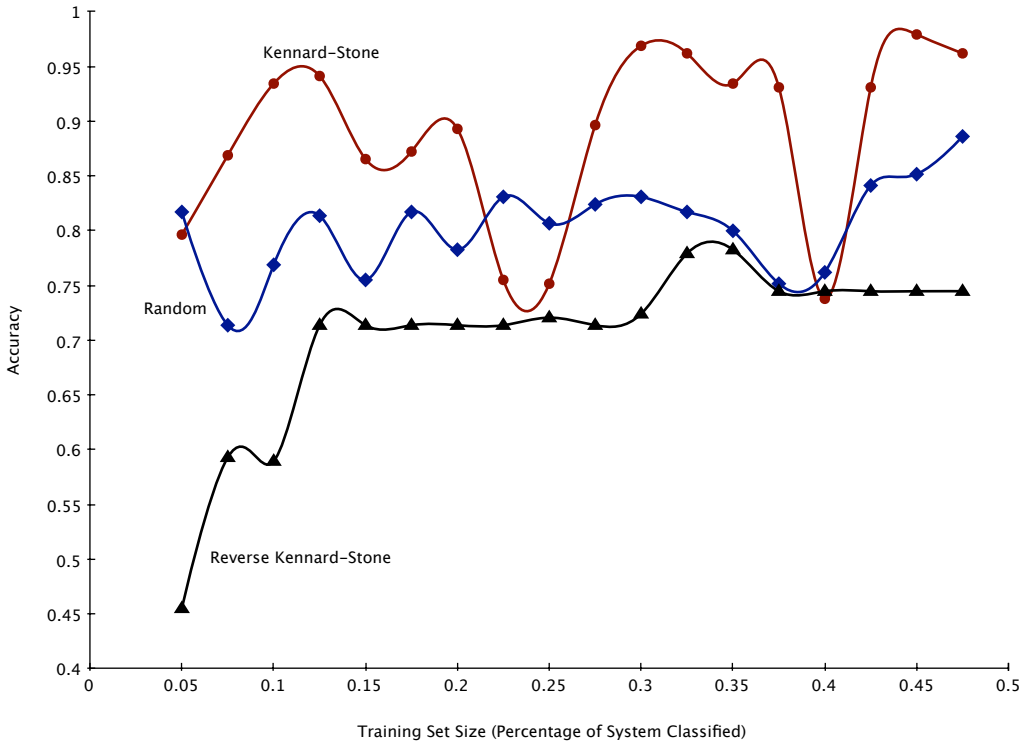

Figure 4.7: Learning Curve for ID3 on Scarab

in the learning curves generated for the ID3 numerical algorithm on the Scarab data set. This confirms our suspicion that the numerical properties of the ID3 numerical algorithm make it inappropriate for this particular problem due to the continuous numerical data.

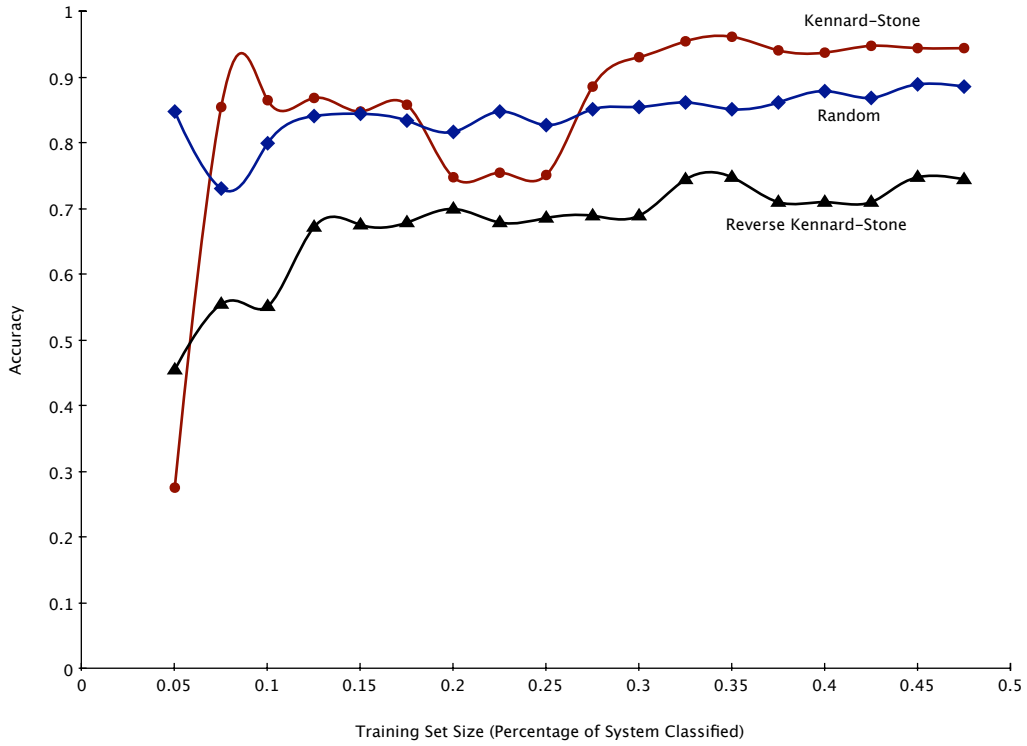

Figure 4.8: Learning Curve for Vote on Scarab

Figure 4.8 shows the learning curves for the Vote algorithm executed on the Scarab dataset for each of the training set selection methods. As with the Panda data set we are averaging the results of KNN, SVM, and ID3 via a majority rules vote. However, it is interesting here that now the Kennard-Stone selection method achieves the highest performance levels on all but four data points.

Recommendations Based on Learning Curves

For training set selection Reverse Kennard-Stone may be interesting for experimental comparison but overall results show that it is an ineffective method of optimizing training set selection. Indeed, random selection produces learning curves with less variance across training set sizes. In practice this translates to greater consistency than the other selection methods. Based on the work of Wu et. al. [57] we know that it is possible to find optimal training set selectors for some data sets.

Kennard-Stone exceeds the performance of random selection on the Scarab data set. However, there is not enough evidence to determine if this would be the case in general for any given data set.

Based on the learning curves both KNN and SVM seem to be appropriate learning methods for this problem. KNN does have a clear advantage in the latter part of the Kennard-Stone learning curve for Scarab, and maintains a slight performance advantage with random selection as well. Given the variances encountered in the Kennard-Stone learning curves it is difficult to make a recommendation based on that data. Since KNN is simpler to implement it may have an advantage there but SVM should not be ignored as there may be potential for further improvements. ID3 with numerical binning is not a good choice given the variance of the learning curve as there is no way to determine if we have trained with enough data to ensure adequate performance for our needs.

The performance of an appropriate learning algorithm given a randomly selected training set size of between 7.5% and 10% on a sufficiently large system is not dramatically improved by further classifying the system. We can see this is true for Scarab when using either KNN or SVM.

4.3 External Validation

In order to validate the approach we want to determine how another engineer would classify a system, and compare his classification to our own. To accomplish this we asked a student in our computer science program to examine the Panda system and provide a classification. We explained that core concepts are those concepts that describe the domain of the system, and we simply want to identify the classes which are representative of core concepts. We did not answer questions about the software but did give a brief demo using the software. We also gave a brief introduction to the domain of the software. Panda is a proof assistant, and the student was not familiar with logical proofs of this form. The student was aware that we were attempting to evaluate our approach but did not have access to our existing classification.

Comparing the results of our student's classification versus our classification of Panda, there were 12 classifications on which we do not agree out of the 90 classes in the system. We examine each of the differences to determine why the discrepancies exist.

The first set of discrepancies exist in four classes called *AbstractCommand*, *CommandManager*, *CommandRedo*, and *CommandUndo*. We originally classified these as concepts, however our student classified them as non-concepts. The Panda system implements a large set of classes that represent use cases a user would want to execute in the logical proof domain. Most of these classes represent domain specific use cases and so are clearly concepts. However, *CommandManager* manages a command history. This is a convenience for the user to allow undo and redo of commands previously executed, but are not concepts in the logical proof domain. The *CommandRedo* and *CommandUndo* classes represent the redo and undo use cases that are not concepts in the logical proof domain. The *AbstractCommand* class is the base class for all commands, however it does not include any domain specific knowledge and therefore is not a concept in the logical proof domain. So, in this case we made a mistake in our initial classification that our student did not.

The second set of discrepancies exist in seven classes called *FormulaPanel*, *InsertionPoint*, *PanelGetAssumptions*, *PanelGetConclusion*, *PanelPromptForLine*, *PanelSelectCommand*, and *PanelWhichEquation*. Each of these classes model GUI based user interactions with the logical proof domain, but do not contain domain specific knowledge. We did not classify these as concepts though our student did. Our student made a mistake in this classification that we think is related to the use of domain terms in the class names. A similar class in this set *PanelGetInputLine* does not include domain terminology in the name and was correctly classified by the student as a non-concept.

The final discrepancy is the *scanner* class. This class is an implementation of a tokenizer for the Panda specific logical proof language. Our student mistakenly classified this as a domain concept.

Overall there was strong agreement between our classification and our student's classification. In fact 86.6% of our responses where the same. Interestingly enough this is on the order of the same performance as the classifiers we have built. In other words our classifiers are accurate to the same level as two engineers' opinions of the *correct* classification.

Table 4.10 on page 62 shows the results of correcting the data set classifications on *AbstractCommand*, *CommandManager*, *CommandRedo*, and *CommandUndo*. We see our original

Table 4.10: Summary of Average Accuracy for Preliminary versus Corrected Panda Results

Experiment	Average Accuracy	Training Set Size	Classes in System
1 Panda Default	69.39%	35	90
1-FSC Panda Default	73.53%	35	90
2 Panda KNN	73.54%	35	90
2-FS Panda KNN	88.41%	35	90
2-FSC Panda KNN	94.29%	35	90
3 Panda SVM	75.65%	35	90
3-FS Panda SVM	89.29%	35	90
3-FSC Panda SVM	93.49%	35	90
4 Panda ID3	81.09%	35	90
4-FS Panda ID3	84.21%	35	90
4-FSC Panda ID3	88.57%	35	90
5 Panda ID3 + Ada Boost	78.91%	35	90
5-FS Panda ID3 + Ada Boost	82.86%	35	90
5-FSC Panda ID3 + Ada Boost	88.49%	35	90
6 Panda Vote	73.54%	35	90
6-FS Panda Vote	87.75%	35	90
6-FSC Panda Vote	95.24%	35	90

preliminary results versus the results with feature selection and correction of the classification errors found in external validation marked with FSC. We have also included the results after feature selection, marked with FS, for comparison. For KNN and SVM the feature selection had more affect on the accuracy than the correction of classification errors. The ID3 numerical based experiments were more affected by the error corrections than by the feature selection though the increases in accuracy were on the same order as those noticed for KNN and SVM so we are seeing that they were less improved by feature selection not more improved by error correction.

In Table 4.11 the difference in precision and recall are shown between the initial experiments as discussed in Section 3.5, the feature selection experiments, and the validation experiments performed after correction of the Panda data set. The complete confusion matrices for each post validation experiment performed are found in Appendix B. If any doubts remain regarding the ability of a machine to solve the concept classification problem the results here should eliminate them. KNN has over 90% recall on both the concept and non-concept classes. While the results for

Table 4.11: Post Validation Precision and Recall

	Recall		Precision	
Experiment	true Concept	true Non-Concept	predicted Concept	predicted Non-Concept
2 Panda KNN	62.56%	80.57%	59.08%	82.76%
2-FS Panda KNN	80.26%	92.07%	81.94%	91.23%
2-FSC Panda KNN	91.82%	95.16%	87.07%	97.04%
3 Panda SVM	33.59%	99.08%	94.24%	76.90%
3-FS Panda SVM	71.54%	97.24%	92.08%	88.40%
3-FSC Panda SVM	79.39%	98.49%	94.93%	93.09%
4 Panda ID3	73.59%	86.32%	70.69%	87.94%
4-FS Panda ID3	73.33%	89.08%	75.07%	88.17%
4-FSC Panda ID3	76.36%	92.90%	79.25%	91.72%
5 Panda ID3 + Ada Boost	81.79%	66.78%	52.47%	89.11%
5-FS Panda ID3 + Ada Boost	73.85%	86.90%	71.64%	88.11%
5-FSC Panda ID3 + Ada Boost	82.73%	90.54%	75.62%	93.66%
6 Panda Vote	58.21%	96.09%	86.97%	83.68%
6-FS Panda Vote	78.21%	94.83%	87.14%	90.66%
6-FSC Panda Vote	88.48%	97.63%	92.99%	95.98%

KNN are the most accurate, each of the other machine learning algorithms has also performed well enough to support the hypothesis.

This set of experiments provides further evidence to support our hypothesis that we have identified a set of metrics that serve to detect core domain concepts in source code. Looking just at KNN we see an improvement that averaged 5.29 more correct classifications based on only 4 corrections implying that the algorithm has learned from those corrections.

Chapter 5

Limits of Approach

In this chapter we discuss the limits of the approach with respect to composite concepts. We identify the types of composite concepts and discuss the limits of our approach with each type.

5.1 Composite Concepts

We define a composite concept as a domain concept from an idealized representation of the domain for which there is no one-to-one mapping to a class in the software system. We consider two possible scenarios based on one-to-many relationships. First, we have the situation where multiple concepts map to one class. Second, we have the situation where one concept maps to multiple classes.

Separation of concerns is a software design principle that states each module of a software system should address a separate concern [17]. We might think of multiple concepts mapping to one class as not having correct separation of concerns. For example, when modeling the concept of a car that has a relationship with the concept of an engine, our software system might have been implemented as a single car class perhaps we are modeling traffic patterns or the movement of people. Perhaps our car class has a method that increases revs via acceleration for example, however this is a method that could be implemented quite differently if further details of the car were modeled. Given the level of detail needed to implement the software requirements there is no need to implement an engine class. Consider also the example of fees calculated for a shopping cart application. Possible fees include taxes and shipping charges. This shopping cart application has no knowledge of either taxes or shipping charges but instead only the abstract concept of a fee. A separate external software system is responsible for the calculation of fees and provides an interface to the shopping cart application to retrieve fees for a given cart. Each fee consists of an amount in a given currency and a label to present to the shopping cart user. So in this example we have a domain that includes concepts of taxes and shipping charges but these concepts do not exist in the shopping cart application. Instead there is only the abstract concept of a fee. We call this a

type 1 composite concept and we expect to encounter these whenever software is modeling an abstraction.

The situation where one concept maps to multiple classes would primarily appear to be a theoretical construct as we have not actually witnessed such entities in the real world. The closest situation observed is two classes that are very tightly coupled. However, even in this case it does not appear that a single concept is being represented by the combination but rather each class represents two distinct yet tightly bound concepts. However, for thoroughness we show that it is computationally difficult to identify such entities in the general case. If more specific information were known about what these entities look like if they do exist in real world implementations then it may be possible to optimize identification based on the constraints found. We call this a type 2 composite concept.

The idea of a composite concept is based on the assumptions that there exists some idealized domain for which there should exist some one-to-one mapping to the implementation of those concepts, and that it is possible to incorrectly map the domain to the implementation. This seems to be inaccurate in at least one respect. First, we are working to identify a domain representation from a given implementation; this is like drawing a map from a given territory. The idea that the domain representation we recover from software is incorrect compared to some idealized domain is like saying that the map we have created by measuring the territory is incorrect because it does not appear as we expected prior to measuring. In this case we are simply discovering that the territory is not what we expected. Second, as Albert Korzybski said “the map is not the territory” [33] meaning that the model is not the thing, or in our case the domain is not the implementation. We should not forget that the domain, regardless of how formally it is defined, is simply a model of reality and that the reality being modeled is the software implementation. The difficulty of refining a model to accurately depict the thing being modeled is humorously addressed by a character of Lewis Carroll’s in saying “we now use the country itself, as its own map, and I assure you it does nearly as well.” [14] Since our approach bases the model on the reality encountered in implemented software we expect it to do “nearly as well” as any model derived prior to understanding the reality of the system. So from the perspective of formal

modeling we have identified limits to this approach that we might like to overcome. However, from a pragmatic perspective interested in modeling the reality of the software as implemented we have identified an important property of the approach in that it can only detect the structures present in the software and not fantasies that exist only in the minds of the observer.

5.2 Type 1 Composite Concepts

Consider a thought experiment based on a graph where each vertex is a class in the system, and each edge is a relationship between classes. Then very little information is lost in the translation of the ideal representation to the system when mapping two concepts to one class. The classification approach we use is primarily dependent on dependency relationships between classes as represented by the USES and USED-BY metrics. In an ideal representation of the domain concepts we would see two vertices representing the two concepts in question but in our implementation we find only one vertex representing both concepts. Consider the transformation necessary to merge two vertices from our ideal representation to a single vertex. The effect on the edges in the system is limited to removing edges between the two vertices, and replacing the the two ideal representations in all other relationships with the new composite concept. Any edges representing dependencies outside the two affected vertices remain such that we have merged the dependencies of the two vertices into a single new vertex. This represents little loss of information in the metrics we are observing. This, along with a few examples, could form an argument for explaining why our system is able to detect type 1 composite concepts. However, we should first examine the idea of the ideal representation.

We consider that there is no ideal representation that works for every situation in every system we are modeling. Looking at our car example above we might say that a single class representation is equally valid if all we are concerned about in our system is transportation of a few people. In this case we could easily argue that the correct representation of the car at the domain level abstracts away the concept of engine, tires, etc. Let us not forget that the domain we are trying to recover is the model which the developers of our software were working to implement. We can conclude that every concept in our domain is some level of abstraction of some other composition of concepts

that we have chosen to ignore in our particular domain, and therefor every domain concept we are detecting in our experiments is at some level of abstraction a type 1 composite concept.

So we have the question "Is our approach limited such that these type 1 composite concepts are not recognized or are more difficult to recognize?" The answer is if these limits to our approach existed we would not find accurate results for any class since we have shown that any particular concept can be broken down as a composite concept at a different level of abstraction.

5.3 Type 2 Composite Concepts

We formally define a composition of classes as a type 2 composite concept if there exists a relationship between two or more classes in the system S such that together they form a domain concept but individually they have no corresponding domain concept. Given this mapping we could define a functional notation $f_c(P_i)$ which maps the partition $P_i \subset S$ to boolean values, such that if all the classes in P_i form a composite concept then f_c is 1, and f_c is 0 otherwise. Let us assume that we can use this mapping to develop a classifier $classifier(P_i)$ to return a pair $(label, confidence)$. Given this classifier we would like to classify the complete system such that for every class and partition of classes forming a composite concept we have minimized the error in the system, or maximized the confidence.

Naively, we can construct a brute force implementation of this algorithm which looks at all of the combinations of classes, computes the confidence on those classifications, and eventually gives us the highest confidence solution. However, ideally we would be able to construct a solution that runs in polynomial time so that we might solve reasonably large problems of this nature. Unfortunately we are unable to construct such an algorithm as this problem is NP-complete.

Definition 2 (BIN-PACK algorithm)**.** BIN-PACK(A,V) calculates the number of bins B of size V needed to hold the $A = a_1, ..., a_n$ items, also finds the B-partitioning of $S = 1, ..., n$ that results.

Definition 3 (Algorithm for optimal partitioning of concept composition)**.** COMPOSITE-CONCEPT(S,ω,k) calculates the partitioning of S such that ω of those partitions is less than or equal to the constant k. Each element of S must appear in one and only one partition.

Theorem 1. COMPOSITE-CONCEPT *is in NP.*

Proof. The following is a verifier V for COMPOSITE-CONCEPT.

V = "On input $\langle\langle S, \omega, k\rangle, P_0, \ldots, P_n\rangle$:

1. Test whether $\omega(P_i) \leq k$ for each $P_i \subset S$.
2. Test whether $P_x \bigcap_n P_y = \emptyset$ for $x, y \in \{0, \ldots, n\} \wedge x \neq y$.
3. Test whether $\bigcup_{i=0} P_i = S$.
4. If all three pass, *accept*; otherwise, *reject*."

□

Theorem 2. COMPOSITE-CONCEPT *is NP-complete.*

Proof. We already know that COMPOSITE-CONCEPT $\in$ NP, so we now show that BIN-PACK $\leq_P$ COMPOSITE-CONCEPT. In other words BIN-PACK is polynomial time reducible to COMPOSITE-CONCEPT.

Given BIN-PACK$\langle A, v\rangle$ we find a mapping to COMPOSITE-CONCEPT$\langle S, \omega, k\rangle$.

Let us define S, the set of classes, given A, the set of items in our BIN-PACK problem, as follows:

$$S = \{f(x) | x \in A\} \tag{5.1}$$

or equivalently:

$$S = f(A) \tag{5.2}$$

where $f(x)$ is a function mapping the items in A to classes in S, and $f^{-1}(x)$ is the inverse. The set S is easily computed given A in polynomial time on the order of $O(n)$ where n is the cardinality of A.

Let us define ω as follows:

$$\omega(P_i) = \sum_{x \in P_i} f^{-1}(x) \tag{5.3}$$

which states that ω has the value of the sum of the size values for each item in the partition. The function ω is independent of the input to BIN-PACK and so this step of the transformation is

computed in constant time, because no part of the transformation regarding ω will change based on input.

Let us define $k = v$. This assignment is a constant time step in the transformation of BIN-PACK to COMPOSITE-CONCEPT.

Now executing an algorithm that solves COMPOSITE-CONCEPT results in a partitioning of S such that ω for each partition is less than or equal to k, or tells us that no such solution exists. Given the mapping above, either P_i contains a set of classes whose integer properties f^{-1} sums to less than v resulting in a partitioning of S that satisfies the constraint $k = v$, or there is no subset of S that can be placed in that partition to meet those restraints. If the constraints are met then we find the answer to BIN-PACK is the sets $f^{-1}(P_i)$ where $i \in \{1,...,B\}$, because $\omega(P_i) \leq k$ therefore $\sum_{x \in P_i} f^{-1}(x) \leq k$ and $k = v$ which implies that $\sum_{a \in A_i} a \leq v$ precisely the condition for a solution to BIN-PACK.

Since, we can reduce BIN-PACK to COMPOSITE-CONCEPT in a series of steps that are polynomial time and COMPOSITE-CONCEPT is in NP, COMPOSITE-CONCEPT is NP-complete because BIN-PACK is NP-complete. □

Theorem 2 shows that using our technique it is not practically possible to detect type 2 composite concepts. Our technique looks for an optimal match of a composite concept among many potential composite concepts, since we assume the definition of our detection function can produce a spectrum of values for "compositeness". However, some other technique might define a simple decision algorithm to the detection of composite concepts. We would assume the this algorithm would take a subset of classes in the system and return either true if that subset is a composition of classes which represents a concept in the domain, or false otherwise. We now show that this technique is also limited in detecting type 2 composite concepts by the impracticality of solving NP-complete problems.

Definition 4 (Algorithm for detection of concept composition)**.**
COMPOSITE-CONCEPT-DECISION(S,α) determines if there exists a subset of S such that the subset includes classes which make up a composite concept.

Theorem 3. COMPOSITE-CONCEPT-DECISION *is in NP.*

Proof. The following is a verifier V for COMPOSITE-CONCEPT-DECISION.

V = "On input $\langle\langle S, \alpha\rangle, P\rangle$:

1. Test whether $\alpha(P)$ is true.
2. Test whether $P \subset S$.
3. If both pass, *accept*; otherwise, *reject*."

□

Theorem 4. COMPOSITE-CONCEPT-DECISION *is NP-complete.*

Proof. We already know that COMPOSITE-CONCEPT-DECISION $\in$ NP, so we now show that SUBSET-SUM $\leq_P$ COMPOSITE-CONCEPT-DECISION. In other words SUBSET-SUM is polynomial time reducible to COMPOSITE-CONCEPT-DECISION.

Given SUBSET-SUM$\langle T, t\rangle$ we find a mapping to COMPOSITE-CONCEPT-DECISION$\langle S, \alpha\rangle$.

Define S, the set of classes, given T, the set of integers in our SUBSET-SUM problem, as follows:

$$S = \{f(x) | x \in T\} \tag{5.4}$$

or equivalently:

$$S = f(T) \tag{5.5}$$

where $f(x)$ is a function mapping the integers in T to classes in S, and $f^{-1}(x)$ is the inverse. The set S is easily computed given T in $O(n)$ time where n is the cardinality of T.

Define α as follows:

$$\alpha(P) = true \text{ iff} \sum_{x \in P} f^{-1}(x) = t \tag{5.6}$$

so that $\alpha(P)$ is true if and only if the sum of $f^{-1}(x)$ where $x \in P$ is equal to t. The function α is independent of the input to SUBSET-SUM and so this step of the transformation is computed in constant time, because no part of the transformation regarding α will change based on input.

Now executing an algorithm that solves COMPOSITE-CONCEPT-DECISION results in the detection of P such that α is true for P, or tells us that no such subset exists. Given the mapping above, either P contains a set of classes whose integer properties f sums to t resulting in a subset of S that satisfies the constraint such that α is true, or there is no subset of S for which α is true. If the constraints are met then we find the answer to SUBSET-SUM is the set $f^{-1}(P)$, because in that situation $\alpha(P) = true$ therefore $\sum_{x \in P} f^{-1}(x) = t$ precisely the condition for a solution to SUBSET-SUM.

Since, we can reduce SUBSET-SUM to COMPOSITE-CONCEPT-DECISION in a series of steps that are polynomial time and COMPOSITE-CONCEPT-DECISION is in NP, COMPOSITE-CONCEPT-DECISION is NP-complete because SUBSET-SUM is NP-complete. □

So, finding composite concepts is a fundamentally difficult problem regardless of the technique that is used to detect those composite concepts. In Section 1.2 we ask "Is it possible to use this approach to classify composite concepts made up of more than one class, which individually are not concepts?" Given this result we find that it is not possible to use this or any other approach to optimally classify type 2 composite concepts in a reasonable amount of time.

Chapter 6

Conclusion

The primary contribution of this thesis is showing that concept identification using a machine learning based classification approach is possible. Prior work in the field does not attempt concept identification via machine learning classification. In the following sections we discuss our contributions and findings, suggest areas for future work, and finally wrap up with a discussion of the significance of the work in this field.

6.1 Contributions and Findings

Our contributions include the following:

1. We have shown that accurate concept identification is possible using machine generated classifiers.
2. We discovered the optimal set of metrics to include in the feature set, and defined the USES and USEDBY metrics.
3. We have shown that a training set size of 7.5 to 10% of the system can yield results nearly as accurate as training sets using half the system.
4. We proved that problems involving detection of concepts crossing class boundaries are NP-Complete.

In this paper we have covered a lot of ground. So it is interesting to look back and see where we started and where the results have taken us along with what we developed to get from the initial idea to the completed work.

We began with a simple informal question “Is it possible to identify concepts based on metrics using SVM?” Our initial results were a proof of concept. We found an open source solution that would generate metrics from Java code. We manually classified Panda, and then generated a data set from the combination of the metrics and the classifications. After some manual formatting of the data we were able to import it into MATLAB and run a SVM classifier. We did some testing similar to that found in Section 3.5 and found that there was a statistically significant result.

At this point the real work began. There were so many questions to answer, which are formalized in Section 1.2. We found ourselves skeptical of the initial results but intrigued by the implication that we could identify concepts using a machine learning approach. There were a lot of challenges to overcome.

Our initial results were statistically significant, but there can be a large gap between statistically significant and practically useful. We had achieved average accuracy as high as 70%, but we would need to increase performance to have a shot at a pragmatic tool. There were multiple paths to explore.

We would need to examine the feature set and determine what we needed to add and what should be removed to maximize performance, but at the same time we did not want to have an adverse effect on our analysis of learning algorithms. We developed a new metrics collector. The open source tool we had used initially was not suited to the task of generating machine readable files, and we knew we would want to add additional metrics. Initial results showed a strong reliance on metrics related to the structure of the software, for example DIT. Our initial set of metrics did not include USES or USEDBY so we added these to our metrics collector in the hope of improving accuracy. This turned out to be a good decision. As discussed in Section 4.1, USEDBY and DIT were the two common metrics selected by each of the three feature selection algorithms. Together USEDBY and DIT are more than capable of producing statistically significant results.

We wanted to examine different learning algorithms to determine if superior performance is possible with an algorithm other than SVM. The number of experiments we wanted to perform would grow with each additional learning algorithm we decided to examine. We needed a better method of collecting data sets. MATLAB had a disadvantage because each of the learning algorithms implemented in MATLAB would take different inputs. We were concerned that producing an automated transformation of the common data set format to the input format for each algorithm could introduce errors in the transformation phase. If we found differences in algorithms it would be challenging to ensure that those originated in the algorithm and not in the automated transformation of the data set. Our search for a tool with common inputs yielded RapidMiner. RapidMiner accepts input from XRFF and implements many of the machine learning algorithms of

interest. XRFF is an XML based file for storing machine learning data sets. To speed the process of creating XRFF we developed an Eclipse plugin that would use our metrics collector and annotations on Java source files to generate the required XRFF file. We developed the annotations to allow adding classification information directly to Java source code so that the source would be the authority for all date necessary to produce and test classifiers.

We needed to minimize the size of training sets to ensure a useful approach in practice, but we did not want to artificially introduce a limit on size that some use cases might comfortably exceed. In order to map out the effects of different sized training sets, changes in training set selection methods, and determine effectiveness of learning algorithms we produced learning curves for each of the different selection methods and learning algorithms. A learning curve shows the relationship between performance and the size of a training set. This allowed us to determine that a training set of 7.5 to 10% is nearly as effective as a training set representing 50% of the system. We showed random selection is the most consistent and effective means of selecting a training set. We also found that KNN was the most consistent performer among the learning algorithms we tested.

Finally, we needed to know the limits of this approach. How would the structure of concepts encoded in software systems affect our ability to detect them using classifiers? We discussed two possible structures besides a one to one mapping of domain knowledge to implementation. Classes that represent more than one concept are simple to detect using our approach. We discussed what we call Type 1 composite concepts in Section 5.2 and showed that these are concepts at differing levels of abstraction. Almost any concept can be further refined into a set of related concepts, but the correct level of abstraction is determined by the domain that is actually being implemented. We discussed Type 2 composite concepts in Section 5.3. Type 2 composite concepts represent a domain concept implemented by more than one class. These Type 2 composite concepts probably do not exist, and as we have shown are difficult to detect due to NP-Completeness of the problem for any classification algorithm including our approach.

6.2 Future Research

While exploring this topic we have discovered additional questions that could be good topics for future exploration.

Understanding why USEDBY and DIT are good indicators of domain concepts may indicate other metrics that we have yet to consider. Currently the structural metrics, like USEDBY, and USES, only count generic dependencies. Perhaps separating dependencies into more specific relationships could provide further improvements in accuracy. For example, composition and delegation could be tracked separately.

There is potential to improve on the training set sizes in future work. Reduction of the size of training sets needed to attain appropriate accuracy make the approach more efficient for an engineer to utilize in practice. We investigated training sets that are selected randomly, by the Kennard-Stone method, and with the reverse Kennard-Stone method, but perhaps there is a better way to go about this training process. It might be interesting to examine a combination of Kennard-Stone with random selection.

We have explored several different learning algorithms representative of several categories of learning algorithms but still more exist that we have not tested. Since we already see variance in the abilities of different algorithms it is reasonable to test additional algorithms to determine if superior performance can be achieved on the classification problem.

6.3 Significance

With out the introduction of new tools and techniques software engineers will continue to struggle with identifying the important domain knowledge contained in existing software systems resulting in higher development costs. Fred Brooks famously asserted that maintenance accounts for 90% of software costs [10]. While an exact accounting cannot be generalized so succinctly most practicing engineers would agree to the qualitative point that maintenance is the most expensive aspect of software development. The expense of maintaining existing software can be roughly divided into two categories; discovery of the existing system, and executing the required changes. It is the discovery process that we hope to accelerate via use of our approach. Tools that assist in directing study of the system toward domain concepts speed the engineer's comprehension tasks allowing him to make needed changes at a quicker pace.

Internal documentation artifacts, those meant to capture the tribal knowledge of the developers, are notoriously outdated but are meant to assist in the discovery process. In practice documentation

rarely exists, and when it does it may be a snapshot of an earlier system that has evolved out of existence. The benefits of using these outdated documents can be overwhelmed by the misinformation that only applied to an earlier version of the system. Alone this does not account for the necessity of our approach; other approaches might work equally well in an environment with outdated or partially complete documentation particularly those that do matching of components of the software to knowledge contained in documents.

In an industry where the difference between market leadership and runner up can be measured not just in profits but the survival of a business it could be argued that time spent on internal documentation results in a competitive disadvantage. The creative work of the engineer is in the software that is developed. Time spent documenting is time not spent creating the next software release. Allocation of scarce creative resources to document software could be considered misallocation of funds given the opportunity loss represented by not delivering new software features. This would be an especially egregious mistake if it were possible to automate the recovery of that documentation at a later date. It is in this hyper-competitive environment that our approach excels.

When we frame internal documentation as an artifact with greatest value to future development, then we see that we are paying up front in effort invested for documentation that has an unknown value later. Our approach to identifying domain knowledge from the source is more suited to this environment where we can chose to pay the cost of learning about the domain at the time we have found a need to modify the software. In this respect our approach is unique in application when compared to competitive approaches.

REFERENCES

[1] David W. Aha, Dennis Kibler, and Marc K. Albert. Noise-tolerant instance-based learning algorithms. In *Instance-Based Learning Algorithms 65*, pages 794–799. Morgan Kaufmann, 1989.

[2] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Mach. Learn.*, 6(1):37–66, 1991.

[3] Apache Software Foundation. Torque. [Online] Available http://db.apache.org/torque/.

[4] Apache Software Foundation. Turbine. [Online] Available http://jakarta.apache.org/turbine/.

[5] Ted J. Biggerstaff. Design recovery for maintenance and reuse. *Computer*, 22(7):36–49, 1989.

[6] Ted J. Biggerstaff, Bharat G. Mitbander, and Dallas Webster. The concept assignment problem in program understanding. In *ICSE '93: Proceedings of the 15th international conference on Software Engineering*, pages 482–498, Los Alamitos, CA, USA, 1993. IEEE Computer Society Press.

[7] Ted J. Biggerstaff, Bharat G. Mitbander, and Dallas E. Webster. Program understanding and the concept assignment problem. *Communications of the ACM*, 37(5):72–82, 1994.

[8] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM Press.

[9] James F. Bowring, James M. Rehg, and Mary Jean Harrold. Active learning for automatic classification of software behavior. In *ISSTA '04: Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis*, pages 195–205, New York, NY, USA, 2004. ACM.

[10] Frederick P. Brooks, Jr. *The Mythical Man-Month: Essays on Softw*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1978.

[11] Yuriy Brun and Michael D. Ernst. Finding latent code errors via machine learning over program executions. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 480–490, Washington, DC, USA, 2004. IEEE Computer Society.

[12] Bugzilla Organization. bugzilla.org. [Online] Available http://www.bugzilla.org/.

[13] Maurice M. Carey and Gerald C. Gannod. Recovering concepts from source code with automated concept identification. In *ICPC '07: Proceedings of the 15th IEEE International Conference on Program Comprehension*, pages 27–36, Washington, DC, USA, 2007. IEEE Computer Society.

[14] Lewis Carroll. *Sylvie and Bruno Concluded*. Macmillan and Co., 1893.

[15] Paul R. Cohen. *Empirical Methods for Artificial Intelligence*. MIT Press, 1995.

[16] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[17] Edsger Wybe Dijkstra. *Selected Writings on Computing: A Personal Perspective*. Springer-Verlag, New York, NY, USA, 1982.

[18] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA, 1998. ACM Press.

[19] Eclipse Organization. Eclipse. [Online] Available http://www.eclipse.org.

[20] Alexander Egyed. Automated abstraction of class diagrams. *ACM Trans. Softw. Eng. Methodol.*, 11(4):449–491, 2002.

[21] J. M. Favre. Cacophony: metamodel-driven software architecture reconstruction. In *WCRE 2004: Proceedings of the 11th Working Conference on Reverse Engineering*, pages 204–213, 2004.

[22] Haibo He and E.A. Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263 –1284, sept. 2009.

[23] Brian Henderson-Sellers. Modularization and mccabe's cyclomatic complexity. *Communications of the ACM*, 35(12):17–19, December 1992.

[24] Brian Henderson-Sellers. *A Book of Object Oriented Knowledge*. Prentice Hall, Sydney, Australia, 2nd edition, 1995.

[25] Brian Henderson-Sellers. *Object-Oriented Metrics: Measures of Complexity*. Object-oriented series. Prentice Hall PTR, 1996.

[26] Greg Hodgdon. PANDA : Proof Assistant for Natural Deduction Analysis. Technical Report of MCS Project, Arizona State University, November 2001.

[27] I. Hsi, C. Potts, and M. Moore. Ontological excavation: unearthing the core concepts of the application. In *WCRE 2003: Proceedings of 10th Working Conference on Reverse Engineering*, pages 345–353, 2003.

[28] Andy Hunt and Dave Thomas. Software archaeology. *IEEE Softw.*, 19(2):20–22, 2002.

[29] Ivar Jacobson. *Object-oriented software engineering*. ACM, New York, NY, USA, 1992.

[30] Thorsten Joachims. A statistical learning learning model of text classification for support vector machines. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 128–136, New York, NY, USA, 2001. ACM Press.

[31] Cihan Kaynak. Semantic inconsistency and computational intractability in transitive abstraction rules. Master's thesis, Miami University, August 2008.

[32] R. W. Kennard and L. A. Stone. Computer aided design of experiments. *Technometrics*, 11(1):pp. 137–148, February 1969.

[33] Alfred Korzybski. A non-Aristotelian system and its necessity for rigour in mathematics and physics. In a paper presented before the *American Mathematical Society* at the New Orleans, Louisiana, meeting of the *American Associate of the Advancement of Science*, December 28, 1931. Reprinted in *Science and Sanity*, 1933, p. 747-61.

[34] Jonathan I. Maletic and Andrian Marcus. Supporting program comprehension using semantic and structural information. *Software Engineering, International Conference on*, 0:0103, 2001.

[35] Andrian Marcus. Semantic driven program analysis. *Software Maintenance, IEEE International Conference on*, 0:469–473, 2004.

[36] Andrian Marcus, Andrey Sergeyev, Vaclav Rajlich, and Jonathan I. Maletic. An information retrieval approach to concept location in source code. In *WCRE '04: Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04)*, pages 214–223, Washington, DC, USA, 2004. IEEE Computer Society.

[37] T. Marill and D. Green. On the effectiveness of receptors in recognition systems. *Information Theory, IEEE Transactions on*, 9(1):11 – 17, jan 1963.

[38] Ettore Merlo, Ian McAdam, and Renato De Mori. Feed-forward and recurrent neural networks for source code informal information analysis. *Journal of Software Maintenance*, 15(4):205–244, 2003.

[39] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In Lyle Ungar, Mark Craven, Dimitrios Gunopulos, and Tina Eliassi-Rad, editors, *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA, August 2006. ACM.

[40] George A. Miller. The magical number seven, plus or minus two. *The Psychological Review*, 63:81–97, 1956.

[41] Tom M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

[42] Massimiliano Pontil and Alessandro Verri. Support vector machines for 3d object recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(6):637–646, 1998.

[43] Denys Poshyvanyk, Yann-Gael Gueheneuc, Andrian Marcus, Giuliano Antoniol, and Vaclav Rajlich. Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval. *IEEE Transactions on Software Engineering*, 33(6):420–432, 2007.

[44] Denys Poshyvanyk and Andrian Marcus. Combining formal concept analysis with information retrieval for concept location in source code. *International Conference on Program Comprehension*, 0:37–48, 2007.

[45] Weka Project. Xrff. [Online] Available http://weka.sourceforge.net/wekadoc/index.php/en:XRFF_(3.5.4), January 2007.

[46] Weka Project. Arff. [Online] Available http://weka.sourceforge.net/wekadoc/index.php/en:ARFF_%283.5.1%29, July 2008.

[47] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106.

[48] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[49] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2002.

[50] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.

[51] J. Sklansky and W. Siedlecki. A note on genetic algorithms for large-scale feature selection. In *Handbook Of Pattern Recognition And Computer Vision*, chapter 5, pages 88–107. August 1993.

[52] Davor Svetinovic, Daniel M. Berry, and Michael Godfrey. Concept identification in object-oriented domain analysis: Why some students just don't get it. In *RE '05: Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 189–198, Washington, DC, USA, 2005. IEEE Computer Society.

[53] Tigris. Scarab. [Online] Available http://scarab.tigris.org/.

[54] A. van Deursen, C. Hofmeister, R. Koschke, L. Moonen, and C. Riva. Symphony: view-driven software architecture reconstruction. In *WICSA 2004: Proceedings of Fourth Working IEEE/IFIP Conference on Software Architecture*, pages 122–132, 2004.

[55] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[56] A.W. Whitney. A direct method of nonparametric measurement selection. *Computers, IEEE Transactions on*, C-20(9):1100 – 1103, sept. 1971.

[57] W. Wu, B. Walczak, D.L. Massart, S. Heuerding, F. Erni, I.R. Last, and K.A. Prebble. Artificial neural networks in classification of nir spectral data: Design of the training set. *Chemometrics and Intelligent Laboratory Systems*, 33(1):35 – 46, 1996.

[58] Andy Zaidman, Toon Calders, Serge Demeyer, and Jan Paredaens. Applying webmining techniques to execution traces to support the program comprehension process. In *CSMR '05: Proceedings of the Ninth European Conference on Software Maintenance and Reengineering*, pages 134–142, Washington, DC, USA, 2005. IEEE Computer Society.

APPENDIX A

Post Feature Selection Confusion Matrices

Table A.1 to table A.10 display confusion matrices for the feature selection validation experiments discussed in section 4.1.

Table A.1: Post Feature Selection Precision and Recall for Panda KNN

	actual true	actual false	class precision
predicted true	313	69	81.94%
predicted false	77	801	91.23%
class recall	80.26%	92.07%	

Table A.2: Post Feature Selection Precision and Recall for Panda SVM

	actual true	actual false	class precision
predicted true	279	24	92.08%
predicted false	111	846	88.40%
class recall	71.54%	97.24%	

Table A.3: Post Feature Selection Precision and Recall for Panda ID3

	actual true	actual false	class precision
predicted true	286	95	75.07%
predicted false	104	775	88.17%
class recall	73.33%	89.08%	

Table A.4: Post Feature Selection Precision and Recall for Panda ID3 with Aba Boost

	actual true	actual false	class precision
predicted true	288	114	71.64%
predicted false	102	756	88.11%
class recall	73.85%	86.90%	

Table A.5: Post Feature Selection Precision and Recall for Panda Vote

	actual true	actual false	class precision
predicted true	305	45	87.14%
predicted false	85	825	90.66%
class recall	78.21%	94.83%	

Table A.6: Post Feature Selection Precision and Recall for Scarab KNN

	actual true	actual false	class precision
predicted true	2131	503	80.90%
predicted false	509	5557	91.61%
class recall	80.72%	91.70%	

Table A.7: Post Feature Selection Precision and Recall for Scarab SVM

	actual true	actual false	class precision
predicted true	1876	429	81.39%
predicted false	764	5631	88.05%
class recall	71.06%	92.92%	

Table A.8: Post Feature Selection Precision and Recall for Scarab ID3

	actual true	actual false	class precision
predicted true	1907	614	75.64%
predicted false	733	5446	88.14%
class recall	72.23%	89.87%	

Table A.9: Post Feature Selection Precision and Recall for Scarab ID3 with Ada Boost

	actual true	actual false	class precision
predicted true	2000	717	73.61%
predicted false	640	5343	89.30%
class recall	75.76%	88.17%	

Table A.10: Post Feature Selection Precision and Recall for Scarab Vote

	actual true	actual false	class precision
predicted true	2020	446	81.91%
predicted false	620	5614	90.05%
class recall	76.52%	92.64%	

APPENDIX B

Post Validation Confusion Matrices

Table B.1 to table B.5 display confusion matrices for the external validation experiments discussed in section 4.3.

Table B.1: Post Validation Precision and Recall for Panda KNN

	actual true	actual false	class precision
predicted true	303	45	87.07%
predicted false	27	885	97.04%
class recall	91.82%	95.16%	

Table B.2: Post Validation Precision and Recall for Panda SVM

	actual true	actual false	class precision
predicted true	262	14	94.93%
predicted false	68	916	93.09%
class recall	79.39%	98.49%	

Table B.3: Post Validation Precision and Recall for Panda ID3

	actual true	actual false	class precision
predicted true	252	66	79.25%
predicted false	78	864	91.72%
class recall	76.36%	92.90%	

Table B.4: Post Validation Precision and Recall for Panda ID3 with Aba Boost

	actual true	actual false	class precision
predicted true	273	88	75.62%
predicted false	57	842	93.66%
class recall	82.73%	90.54%	

Table B.5: Post Validation Precision and Recall for Panda Vote

	actual true	actual false	class precision
predicted true	292	22	92.99%
predicted false	38	908	95.98%
class recall	88.48%	97.63%	

APPENDIX C

Data Sets

C.1 Panda

The metrics data set used in this research for the Panda system is shown in Table C.1.

Table C.1: Metrics Data for Panda

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
Panda.-FormulaListElement	true	1.00	0.89	90.00	1.14	51.00	1.07	8.00	14.00	0.00	0.93	0.00	0.00	0.00	0.00	22.00	10.00	16.00
Panda.-PanelGetAssumptions	false	5.00	0.38	57.00	2.00	42.00	1.50	6.00	2.00	0.00	1.00	2.00	0.00	0.00	0.00	2.00	10.00	4.00
Panda.-CommandEquivElim	true	3.00	0.80	56.00	1.67	39.00	1.33	1.00	6.00	3.00	0.83	0.00	0.00	0.00	1.50	2.00	13.00	10.00
Panda.-CommandNotIntro	true	3.00	0.86	75.00	1.50	52.00	1.17	5.00	6.00	3.00	0.83	2.00	0.00	0.00	1.50	2.00	9.00	9.00
Panda.-CommandUndo	false	2.00	0.00	12.00	1.00	3.00	1.00	0.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	2.00	3.00
Panda.-PanelPromptForLine	false	5.00	1.00	30.00	1.00	20.00	1.00	3.00	2.00	0.00	1.50	0.00	0.00	0.00	0.00	11.00	9.00	2.00
Panda.Redo	false	0.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	2.00	0.00	0.00
Panda.-CommandNotEquivIntro	true	3.00	0.91	71.00	2.40	49.00	1.40	6.00	5.00	2.00	0.80	2.00	0.00	0.00	1.20	2.00	8.00	12.00
Panda.-CommandImplIntro	true	3.00	0.87	64.00	1.50	42.00	1.17	4.00	6.00	3.00	0.83	2.00	0.00	0.00	1.50	2.00	9.00	9.00
Panda.-PanelSelectCommand	false	5.00	1.67	28.00	1.00	15.00	1.00	6.00	2.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	9.00	2.00
Panda.-PanelGetInputLine	false	5.00	0.44	78.00	4.50	62.00	2.50	7.00	2.00	0.00	2.00	2.00	0.00	0.00	0.00	5.00	16.00	9.00
Panda.Panda	true	6.00	0.90	225.00	2.67	185.00	1.92	3.00	11.00	0.00	0.17	9.00	1.00	0.00	0.00	1.00	28.00	32.00
Panda.-KeysDialog	false	6.00	0.00	37.00	1.00	32.00	1.00	0.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	8.00	1.00
Panda.-CommandAddAssumption	true	3.00	0.50	18.00	1.00	7.00	1.00	1.00	3.00	0.00	0.67	0.00	0.00	0.00	0.00	2.00	5.00	3.00
Panda.-CommandAndIntro	true	3.00	0.87	70.00	2.00	48.00	1.50	4.00	6.00	3.00	0.83	2.00	0.00	0.00	1.50	2.00	11.00	12.00
Panda.-CommandOrElim	true	3.00	0.83	80.00	1.83	57.00	1.33	5.00	6.00	3.00	0.83	2.00	0.00	0.00	1.50	2.00	10.00	11.00
Panda.-CommandNotEquivElim	true	3.00	0.60	57.00	1.67	40.00	1.33	1.00	6.00	3.00	0.83	0.00	0.00	0.00	1.50	2.00	12.00	10.00
Panda.-PanelGetConclusion	false	5.00	0.33	48.00	2.00	35.00	2.00	5.00	2.00	0.00	1.00	1.00	0.00	0.00	0.00	2.00	10.00	4.00
Panda.Debug	false	1.00	0.60	25.00	1.33	8.00	1.00	0.00	0.00	0.00	0.67	2.00	6.00	0.00	0.00	0.00	1.00	8.00
Panda.-CommandManager	true	1.00	0.44	45.00	2.75	31.00	1.75	3.00	4.00	0.00	0.50	0.00	0.00	0.00	0.00	1.00	4.00	11.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
Panda.scanner	false	1.00	0.75	55.00	4.75	39.00	2.00	4.00	4.00	0.00	0.25	0.00	0.00	0.00	0.00	3.00	5.00	19.00
Panda.sym	false	1.00	0.00	16.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	13.00	0.00	0.00	0.00	0.00	0.00	0.00
Panda.-FormulaPanel	false	5.00	0.00	8.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	2.00	1.00
Panda.-CommandTrustMe	false	3.00	0.75	32.00	1.20	18.00	1.20	1.00	5.00	2.00	0.80	0.00	0.00	0.00	1.20	2.00	8.00	6.00
Panda.-CommandOrIntro	true	3.00	0.83	77.00	1.83	54.00	1.33	4.00	6.00	3.00	0.83	3.00	0.00	0.00	1.50	2.00	13.00	11.00
Panda.-CommandNotElim	true	3.00	0.75	51.00	1.80	36.00	1.20	1.00	5.00	2.00	0.80	0.00	0.00	0.00	1.20	2.00	8.00	9.00
Panda.Counter	false	1.00	0.00	7.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00	1.00
Panda.-CommandRedo	true	1.00	0.00	9.00	1.00	2.00	1.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	2.00
Panda.Formula	true	1.00	0.73	147.00	1.95	110.00	1.23	3.00	17.00	0.00	0.82	0.00	5.00	0.00	0.00	26.00	7.00	43.00
Panda.Undo	false	0.00	0.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	2.00	0.00	0.00
Panda.Prover	true	1.00	0.91	354.00	2.00	176.00	1.42	13.00	19.00	0.00	0.63	12.00	0.00	0.00	0.00	23.00	45.00	38.00
Panda.-Prover$ButtonPanel	false	5.00	0.11	112.00	10.00	86.00	2.00	3.00	2.00	0.00	1.00	16.00	0.00	0.00	0.00	2.00	45.00	20.00
Panda.-FormulaButton	false	6.00	0.00	5.00	1.00	0.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
Panda.-AbstractCommand	true	1.00	2.00	6.00	1.00	2.00	1.00	0.00	2.00	0.00	0.00	1.00	0.00	18.00	0.00	3.00	1.00	2.00
Panda.-PanelWhichEquation	false	5.00	0.50	86.00	4.00	64.00	1.50	8.00	2.00	0.00	3.50	2.00	0.00	0.00	0.00	5.00	15.00	8.00
Panda.-InsertionPoint	false	1.00	0.00	15.00	1.00	10.00	1.00	0.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	2.00	4.00	1.00
Panda.-CommandAndElim	true	3.00	0.60	58.00	1.67	41.00	1.33	1.00	6.00	3.00	0.83	0.00	0.00	0.00	1.50	2.00	12.00	10.00
Panda.-FormulaList	true	5.00	0.89	411.00	2.56	332.00	1.81	2.00	32.00	0.00	0.92	2.00	4.00	0.00	0.00	24.00	21.00	92.00
Panda.-CommandAddConclusion	true	3.00	0.50	18.00	1.00	7.00	1.00	1.00	3.00	0.00	0.67	0.00	0.00	0.00	0.00	2.00	5.00	3.00
Panda.-ProverCommand	true	2.00	0.94	63.00	1.71	42.00	1.29	3.00	7.00	0.00	1.14	0.00	0.00	16.00	0.00	21.00	8.00	12.00
Panda.-CommandEquivIntro	true	3.00	0.88	65.00	2.40	45.00	1.40	4.00	5.00	2.00	0.80	2.00	0.00	0.00	1.20	2.00	8.00	12.00
Panda.-CommandConclude	true	3.00	0.75	48.00	1.80	33.00	1.40	1.00	5.00	2.00	0.80	0.00	0.00	0.00	1.20	2.00	6.00	9.00
Panda.parser	false	2.00	0.98	72.00	1.00	11.00	1.00	1.00	11.00	0.00	0.45	3.00	0.00	0.00	0.00	5.00	10.00	11.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
Panda.-CUP$parser$actions		1.00	1.00	184.00	11.00	170.00	1.50	1.00	2.00	0.00	2.50	0.00	0.00	0.00	0.00	2.00	10.00	22.00
Panda.-CommandImplElim	true	3.00	0.85	66.00	2.60	47.00	1.40	3.00	5.00	2.00	0.80	2.00	0.00	0.00	1.20	2.00	8.00	13.00
Panda.Reason	true	1.00	0.84	67.00	2.40	46.00	1.20	4.00	5.00	1.00	1.60	4.00	0.00	0.00	0.20	18.00	5.00	12.00
java_cup.terminal	false	2.00	0.92	61.00	1.17	24.00	1.00	2.00	8.00	1.00	0.92	5.00	4.00	0.00	0.25	9.00	8.00	14.00
java_cup.-internal_error	false	3.00	0.00	15.00	1.00	5.00	1.00	0.00	2.00	0.00	0.50	0.00	0.00	0.00	0.00	14.00	2.00	2.00
java_cup.-terminal_set	false	1.00	0.75	114.00	1.47	56.00	1.13	1.00	15.00	2.00	0.80	1.00	0.00	0.00	0.13	6.00	6.00	22.00
java_cup.symbol	false	1.00	0.91	28.00	1.22	12.00	1.00	4.00	9.00	1.00	0.33	0.00	0.00	2.00	0.11	12.00	1.00	11.00
java_cup.-lalr_state	false	1.00	0.95	432.00	3.83	374.00	2.17	3.00	18.00	3.00	1.08	3.00	6.00	0.00	0.17	7.00	25.00	92.00
java_cup.-production_part	false	1.00	0.83	36.00	1.71	17.00	1.00	1.00	7.00	3.00	0.43	0.00	0.00	2.00	0.43	10.00	3.00	12.00
java_cup.-parse_action_row	false	1.00	0.75	37.00	3.33	24.00	1.67	2.00	2.00	0.00	0.00	2.00	1.00	0.00	0.00	3.00	2.00	10.00
java_cup.-parse_reduce_table	false	1.00	0.50	43.00	3.00	32.00	2.33	2.00	3.00	1.00	0.00	0.00	0.00	0.00	0.33	3.00	3.00	9.00
java_cup.lexer	false	1.00	0.85	264.00	4.14	216.00	2.07	0.00	1.00	0.00	0.36	12.00	13.00	0.00	0.00	0.00	6.00	58.00
java_cup.-reduce_action	false	2.00	0.83	32.00	1.29	13.00	1.00	1.00	7.00	4.00	0.43	0.00	0.00	0.00	1.14	5.00	6.00	9.00
java_cup.sym	false	1.00	0.00	33.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	30.00	0.00	0.00	0.00	0.00	0.00	0.00
java_cup.-parse_action	false	1.00	1.10	27.00	1.17	8.00	1.00	0.00	6.00	3.00	0.33	4.00	0.00	3.00	0.50	8.00	3.00	7.00
java_cup.-action_part	false	2.00	0.71	33.00	1.25	14.00	1.00	1.00	8.00	3.00	0.50	0.00	0.00	0.00	0.75	4.00	4.00	10.00
java_cup.-lalr_item_set	false	1.00	0.75	160.00	1.68	110.00	1.21	2.00	19.00	3.00	0.63	0.00	0.00	0.00	0.16	2.00	12.00	32.00
java_cup.-nonassoc_action	false	2.00	0.00	26.00	1.17	8.00	1.00	0.00	6.00	4.00	0.33	0.00	0.00	0.00	1.33	1.00	3.00	7.00
java_cup.-parse_action_table	false	1.00	0.50	70.00	2.50	55.00	2.00	2.00	4.00	1.00	0.00	0.00	0.00	0.00	0.25	3.00	6.00	10.00
java_cup.-symbol_set	false	1.00	0.67	100.00	1.88	59.00	1.12	1.00	16.00	3.00	0.69	0.00	0.00	0.00	0.19	2.00	7.00	30.00
java_cup.-symbol_part	false	2.00	0.86	42.00	1.50	20.00	1.00	1.00	8.00	3.00	0.62	0.00	0.00	0.00	0.75	9.00	6.00	12.00
java_cup.-non_terminal	false	2.00	0.96	129.00	1.39	81.00	1.00	4.00	10.00	1.00	0.39	5.00	8.00	0.00	0.20	10.00	10.00	25.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
java_cup.assoc	false	1.00	0.00	7.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00
java_cup.version	false	1.00	0.00	10.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	0.00	0.00	0.00	0.00	1.00	0.00
java_cup.-lalr_transition	false	1.00	0.73	32.00	1.33	16.00	1.00	3.00	6.00	1.00	0.83	0.00	0.00	0.00	0.17	2.00	5.00	8.00
java_cup.Main	false	1.00	0.92	473.00	6.94	388.00	1.71	0.00	1.00	0.00	0.41	32.00	16.00	0.00	0.00	0.00	17.00	118.00
java_cup.-action_production	false	2.00	0.00	16.00	1.00	3.00	1.00	1.00	2.00	0.00	2.50	0.00	0.00	0.00	0.00	1.00	4.00	2.00
java_cup.-lalr_item	false	2.00	0.87	141.00	2.21	97.00	1.36	3.00	14.00	3.00	0.79	0.00	0.00	0.00	0.43	3.00	13.00	31.00
java_cup.-parse_reduce_row	false	1.00	0.50	11.00	1.50	3.00	1.00	1.00	1.00	0.00	0.00	1.00	1.00	0.00	0.00	2.00	1.00	3.00
java_cup.-shift_action	false	2.00	0.83	29.00	1.29	13.00	1.00	1.00	7.00	4.00	0.43	0.00	0.00	0.00	1.14	3.00	6.00	9.00
java_cup.emit	false	1.00	1.01	461.00	4.43	390.00	2.43	0.00	1.00	0.00	2.00	22.00	13.00	0.00	0.00	0.00	19.00	62.00
java_cup.-lr_item_core	false	1.00	0.78	129.00	1.62	83.00	1.12	4.00	16.00	3.00	0.38	0.00	0.00	1.00	0.19	2.00	9.00	26.00
java_cup.-production	false	1.00	0.96	357.00	2.14	273.00	1.31	11.00	31.00	3.00	1.03	2.00	5.00	1.00	0.10	13.00	15.00	77.00
java_cup.parser	false	2.00	0.98	338.00	1.00	18.00	1.00	1.00	15.00	4.00	0.60	3.00	0.00	0.00	0.53	3.00	17.00	15.00
java_cup.-CUP$parser$actions		1.00	0.91	1007.00	19.88	970.00	1.75	11.00	8.00	0.00	1.38	0.00	0.00	0.00	0.00	2.00	17.00	159.00
java_cup.-runtime.lr_parser	false	1.00	0.92	489.00	2.88	391.00	1.62	10.00	39.00	0.00	0.65	1.00	1.00	3.00	0.00	7.00	10.00	115.00
java_cup.-runtime.Symbol	false	1.00	1.00	38.00	1.00	19.00	1.00	6.00	6.00	1.00	2.00	0.00	0.00	0.00	0.17	26.00	3.00	6.00
java_cup.-runtime.-virtual_parse_stack	false	1.00	0.53	48.00	1.83	24.00	1.00	3.00	6.00	0.00	0.33	0.00	0.00	0.00	0.00	1.00	4.00	11.00
java_cup.-runtime.Scanner	false	0.00	0.00	4.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	2.00	0.00	9.00	1.00	1.00
java_cup.-simple_calc.sym	false	1.00	0.00	15.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	12.00	0.00	0.00	0.00	0.00	0.00	0.00
java_cup.-simple_calc.Main	false	1.00	0.00	16.00	3.00	10.00	2.00	0.00	0.00	0.00	1.00	1.00	1.00	0.00	0.00	0.00	5.00	3.00
java_cup.-simple_calc.-scanner	false	1.00	0.75	44.00	5.40	29.00	1.20	2.00	5.00	0.00	0.20	0.00	0.00	0.00	0.00	1.00	4.00	27.00
java_cup.-simple_calc.-parser	false	2.00	0.98	72.00	1.00	11.00	1.00	1.00	11.00	0.00	0.45	3.00	0.00	0.00	0.00	3.00	10.00	11.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
java_cup.-simple_calc.-CUP$parser$actions		1.00	1.00	180.00	8.00	167.00	1.50	1.00	2.00	0.00	2.50	0.00	0.00	0.00	0.00	2.00	10.00	16.00

C.2 Scarab

The metrics data set used in this research for the Scarab system is shown in Table C.2.

Table C.2: Metrics Data for Scarab

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.apache.-fulcrum.-localization.-Localization	false	1.00	0.95	115.00	1.09	36.00	1.09	0.00	0.00	0.00	1.70	1.00	23.00	0.00	0.00	0.00	9.00	25.00
org.tigris.-scarab.actions.-ConfigureIssueList	true	8.00	0.00	77.00	4.50	65.00	3.00	0.00	2.00	1.00	2.00	0.00	0.00	0.00	4.00	0.00	17.00	9.00
org.tigris.-scarab.actions.-DefineXModuleList	true	8.00	0.00	319.00	4.31	262.00	2.62	0.00	13.00	0.00	2.08	0.00	0.00	0.00	0.00	0.00	20.00	56.00
org.tigris.-scarab.actions.-ModifyIssue	true	9.00	0.00	931.00	7.45	833.00	2.85	0.00	19.00	0.00	2.85	0.00	1.00	0.00	0.00	0.00	40.00	149.00
org.tigris.-scarab.actions.-ForgotPassword	true	7.00	0.00	70.00	2.00	55.00	1.67	0.00	2.00	0.00	2.33	0.00	1.00	0.00	0.00	1.00	15.00	6.00
org.tigris.-scarab.actions.-ChangePassword	true	7.00	0.00	70.00	7.00	62.00	4.00	0.00	1.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	11.00	7.00
org.tigris.scarab.-actions.Logout	true	7.00	0.00	29.00	2.00	17.00	2.00	0.00	2.00	1.00	2.00	0.00	0.00	0.00	3.50	0.00	6.00	4.00
org.tigris.scarab.-actions.Redirect	true	8.00	0.00	17.00	1.00	5.00	1.00	0.00	2.00	1.00	2.00	0.00	0.00	0.00	4.00	0.00	5.00	2.00
org.tigris.scarab.-actions.Login	true	7.00	0.00	143.00	5.75	123.00	2.75	0.00	4.00	1.00	2.00	0.00	0.00	0.00	1.75	0.00	17.00	23.00
org.tigris.-scarab.actions.-ScarabSessionValidator	true	5.00	0.00	46.00	7.00	39.00	3.00	0.00	1.00	1.00	1.00	0.00	0.00	0.00	5.00	0.00	10.00	7.00
org.tigris.scarab.-actions.Register	true	7.00	0.00	357.00	6.22	319.00	2.78	0.00	9.00	1.00	1.78	0.00	0.00	0.00	0.78	1.00	21.00	56.00
org.tigris.-scarab.actions.-TemplateList	true	8.00	0.00	339.00	6.38	301.00	3.75	0.00	8.00	2.00	2.25	0.00	0.00	0.00	2.00	0.00	28.00	51.00
org.tigris.-scarab.actions.-HandleRoleRequests	true	8.00	0.00	112.00	3.75	93.00	2.75	0.00	4.00	0.00	2.25	0.00	0.00	0.00	0.00	0.00	18.00	15.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-actions.Search	true	8.00	0.84	939.00	6.42	831.00	3.27	5.00	25.00	2.00	2.08	6.00	1.00	0.00	0.64	0.00	33.00	167.00
org.tigris.-scarab.actions.-SetHomePage	true	8.00	0.00	17.00	2.00	10.00	2.00	0.00	1.00	1.00	2.00	0.00	0.00	0.00	8.00	0.00	5.00	2.00
org.tigris.-scarab.actions.-QueryList	true	8.00	0.00	104.00	3.75	84.00	3.50	0.00	4.00	1.00	2.00	0.00	0.00	0.00	2.00	0.00	17.00	15.00
org.tigris.-scarab.actions.-AssignIssue	true	9.00	0.90	345.00	8.33	316.00	4.33	0.00	6.00	2.00	2.33	2.00	0.00	0.00	3.00	0.00	27.00	50.00
org.tigris.-scarab.actions.-ModifyModule	true	8.00	1.00	164.00	10.00	149.00	5.50	0.00	2.00	0.00	2.00	1.00	0.00	0.00	0.00	0.00	15.00	20.00
org.tigris.-scarab.actions.-MoveIssue	true	9.00	0.00	273.00	14.33	258.00	3.33	0.00	3.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	24.00	43.00
org.tigris.-scarab.actions.-ReportIssue	true	8.00	0.94	638.00	5.61	587.00	2.94	0.00	17.00	1.00	2.56	1.00	1.00	0.00	0.47	0.00	43.00	101.00
org.tigris.-scarab.actions.-ConfigureReport	true	8.00	0.63	936.00	5.45	810.00	3.14	6.00	28.00	0.00	1.90	3.00	1.00	0.00	0.00	0.00	36.00	158.00
org.tigris.-scarab.actions.-ViewIssue	true	8.00	0.00	11.00	1.00	3.00	1.00	0.00	1.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	4.00	1.00
org.tigris.scarab.-actions.admin.-ManageRoles	true	8.00	0.00	121.00	2.33	82.00	1.67	0.00	9.00	2.00	2.11	0.00	0.00	0.00	1.78	0.00	13.00	21.00
org.tigris.scarab.-actions.admin.-ManagePermissions	true	8.00	0.00	70.00	1.67	43.00	1.33	0.00	6.00	2.00	2.00	0.00	0.00	0.00	2.67	0.00	11.00	10.00
org.tigris.scarab.-actions.admin.-Approval	true	8.00	1.00	255.00	11.00	232.00	5.33	0.00	3.00	0.00	1.67	8.00	0.00	0.00	0.00	0.00	26.00	33.00
org.tigris.scarab.-actions.admin.-AppConfigurationSettings	true	8.00	0.00	45.00	8.00	36.00	4.00	0.00	1.00	1.00	2.00	0.00	0.00	0.00	8.00	0.00	11.00	8.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-actions.admin.-ConditionEdit	true	8.00	1.25	92.00	2.60	69.00	1.20	4.00	5.00	2.00	2.00	0.00	0.00	0.00	3.20	0.00	14.00	13.00
org.tigris.scarab.-actions.admin.-ModuleAttributeEdit	true	8.00	0.00	186.00	8.75	164.00	4.00	0.00	4.00	2.00	2.00	0.00	0.00	0.00	4.00	0.00	20.00	35.00
org.tigris.scarab.-actions.admin.-GlobalAttributes	true	8.00	0.00	63.00	2.67	47.00	2.33	0.00	3.00	1.00	2.00	0.00	0.00	0.00	2.67	0.00	13.00	8.00
org.tigris.scarab.-actions.admin.-ArtifactTypeEdit	true	8.00	0.00	403.00	6.33	361.00	3.44	0.00	9.00	1.00	2.00	0.00	0.00	0.00	0.89	0.00	22.00	57.00
org.tigris.scarab.-actions.admin.-AttributeGroupEdit	true	8.00	0.00	482.00	10.88	445.00	3.88	0.00	8.00	1.00	2.00	0.00	0.00	0.00	1.00	0.00	29.00	87.00
org.tigris.scarab.-actions.admin.-GlobalEmailSettings	true	8.00	0.00	28.00	3.00	15.00	2.00	0.00	1.00	1.00	2.00	1.00	0.00	0.00	8.00	0.00	5.00	3.00
org.tigris.scarab.-actions.admin.-ManageUser	true	8.00	0.00	305.00	4.00	262.00	2.40	0.00	10.00	1.00	2.00	0.00	0.00	0.00	0.80	0.00	18.00	40.00
org.tigris.scarab.-actions.admin.-GlobalAttributeEdit	true	8.00	0.00	521.00	9.44	481.00	4.00	0.00	9.00	2.00	2.00	0.00	0.00	0.00	1.78	0.00	26.00	85.00
org.tigris.scarab.-actions.admin.-ManageArtifactTypes	true	8.00	0.00	156.00	5.25	135.00	3.50	0.00	4.00	1.00	2.00	0.00	0.00	0.00	2.00	0.00	17.00	21.00
org.tigris.scarab.-actions.admin.-SetInfoMessage	true	8.00	0.00	20.00	3.00	12.00	2.00	0.00	1.00	1.00	2.00	0.00	0.00	0.00	8.00	0.00	5.00	3.00
org.tigris.scarab.-actions.admin.-GlobalArtifactTypes	true	8.00	0.00	175.00	6.25	155.00	4.25	0.00	4.00	1.00	2.00	0.00	0.00	0.00	2.00	0.00	13.00	25.00
org.tigris.scarab.-actions.admin.-UpdateSearchIndex	true	8.00	0.00	84.00	3.50	50.00	2.50	0.00	2.00	1.00	1.00	3.00	0.00	0.00	4.00	0.00	14.00	7.00
org.tigris.scarab.-actions.admin.-UpdateSearchIndex$UpdateThread		1.00	0.00	20.00	1.50	11.00	1.50	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	14.00	3.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-actions.admin.-GlobalArtifactTypeCreate	true	8.00	0.00	359.00	4.90	312.00	3.10	0.00	10.00	2.00	2.00	0.00	0.00	0.00	1.60	0.00	19.00	49.00
org.tigris.scarab.-actions.admin.-IssueTypeAttributeEdit	true	8.00	0.00	173.00	5.80	146.00	3.20	0.00	5.00	3.00	2.00	0.00	0.00	0.00	4.80	0.00	19.00	29.00
org.tigris.scarab.-actions.base.-BaseModifyIssue	true	8.00	0.00	55.00	5.00	47.00	4.00	0.00	1.00	0.00	2.00	0.00	0.00	3.00	0.00	3.00	11.00	5.00
org.tigris.scarab.-actions.base.-RequireLoginFirstAction	true	7.00	1.04	218.00	1.56	120.00	1.20	0.00	25.00	0.00	1.60	7.00	0.00	34.00	0.00	32.00	16.00	39.00
org.tigris.scarab.-actions.base.-ScarabTemplateAction	true	6.00	1.04	136.00	1.00	49.00	1.00	0.00	22.00	0.00	1.50	5.00	0.00	5.00	0.00	5.00	9.00	22.00
org.tigris.scarab.-actions.setup.-AntRunner	false	1.00	0.95	149.00	3.40	125.00	2.60	2.00	4.00	0.00	1.20	3.00	1.00	0.00	0.00	1.00	19.00	17.00
org.tigris.-scarab.attribute.-StringAttribute	false	5.00	0.00	21.00	3.00	13.00	3.00	0.00	1.00	1.00	1.00	0.00	0.00	2.00	5.00	6.00	5.00	3.00
org.tigris.-scarab.attribute.-DateAttribute	false	6.00	0.50	48.00	3.00	34.00	2.33	0.00	0.00	0.00	2.00	1.00	3.00	0.00	0.00	5.00	8.00	9.00
org.tigris.-scarab.attribute.-TotalVotesAttribute	false	4.00	0.00	39.00	1.12	11.00	1.12	0.00	8.00	0.00	0.38	0.00	0.00	0.00	0.00	2.00	3.00	9.00
org.tigris.-scarab.attribute.-OptionAttribute	false	4.00	0.00	17.00	1.00	3.00	1.00	0.00	3.00	0.00	0.67	0.00	0.00	2.00	0.00	10.00	3.00	3.00
org.tigris.-scarab.attribute.-FreeFormAttribute	false	4.00	0.00	18.00	1.33	14.00	1.33	0.00	3.00	0.00	0.33	0.00	0.00	3.00	0.00	1.00	2.00	4.00
org.tigris.-scarab.attribute.-UserAttribute	false	4.00	0.00	71.00	1.71	45.00	1.71	0.00	7.00	2.00	0.43	0.00	0.00	0.00	1.14	3.00	6.00	12.00
org.tigris.-scarab.attribute.-IntegerAttribute	false	6.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.attribute.-SelectOneAttribute	false	5.00	0.00	7.00	1.00	0.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.attribute.-ComboBoxAttribute	false	6.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
org.tigris.-scarab.cactus.-TestSampleServlet	false	5.00	0.00	22.00	1.00	6.00	1.00	0.00	3.00	0.00	0.50	0.00	1.00	0.00	0.00	1.00	7.00	4.00
org.tigris.-scarab.cactus.-SampleServlet	false	3.00	0.00	9.00	1.00	2.00	1.00	0.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	3.00	1.00
org.tigris.scarab.-components.-TorqueComponent	false	3.00	0.00	25.00	1.00	6.00	1.00	0.00	4.00	3.00	0.50	0.00	0.00	0.00	2.25	0.00	4.00	4.00
org.tigris.scarab.-da.DAFactory	false	1.00	1.00	38.00	2.00	20.00	2.00	0.00	0.00	0.00	0.50	1.00	2.00	0.00	0.00	0.00	9.00	4.00
org.tigris.-scarab.da.-DAFactory$LookupError		4.00	0.00	7.00	1.00	1.00	1.00	0.00	1.00	0.00	2.00	0.00	0.00	0.00	0.00	2.00	9.00	1.00
org.tigris.-scarab.da.-AttributeAccess	false	1.00	0.93	307.00	3.18	244.00	2.27	1.00	11.00	0.00	2.18	6.00	0.00	0.00	0.00	4.00	15.00	35.00
org.tigris.scarab.-da.DAException	false	6.00	0.00	9.00	1.00	1.00	1.00	0.00	1.00	0.00	2.00	0.00	0.00	0.00	0.00	1.00	3.00	1.00
org.tigris.scarab.-feeds.Feed	false	0.00	0.00	5.00	1.00	2.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	2.00	0.00	5.00	1.00	1.00
org.tigris.scarab.-feeds.QueryFeed	true	1.00	0.60	117.00	4.50	100.00	2.50	4.00	4.00	0.00	2.25	1.00	0.00	0.00	0.00	2.00	30.00	18.00
org.tigris.scarab.-feeds.IssueFeed	true	1.00	0.50	52.00	1.33	40.00	1.33	3.00	3.00	0.00	1.33	0.00	0.00	0.00	0.00	2.00	18.00	4.00
org.tigris.-scarab.om.-BaseScarabUserImpl	false	4.00	0.98	3144.00	3.20	2524.00	2.65	36.00	172.00	3.00	0.88	1.00	0.00	1.00	0.07	1.00	25.00	551.00
org.tigris.scarab.-om.Frequency	true	3.00	0.00	6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	2.00	0.00
org.tigris.-scarab.om.-RIssueTypeOptionPeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-ROptionOptionManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-OptionRelationshipManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-BaseActivitySetTypePeer	false	2.00	0.96	395.00	1.55	219.00	1.45	0.00	0.00	0.00	1.34	7.00	38.00	1.00	0.00	1.00	26.00	59.00
org.tigris.-scarab.om.-IssueTypeManager	false	3.00	0.00	43.00	1.17	17.00	1.17	0.00	6.00	2.00	0.50	0.00	0.00	0.00	1.00	1.00	7.00	7.00
org.tigris.-scarab.om.-BaseFrequency	false	2.00	0.91	630.00	2.80	482.00	2.20	7.00	40.00	6.00	0.71	2.00	1.00	1.00	0.30	1.00	17.00	115.00
org.tigris.-scarab.om.-BaseDependPeer	false	2.00	0.92	758.00	2.55	552.00	1.89	0.00	0.00	0.00	1.30	10.00	44.00	1.00	0.00	1.00	30.00	112.00
org.tigris.-scarab.om.-BaseTransitionManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-RModuleUserAttributeManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-ScarabModule	true	4.00	0.97	548.00	2.54	402.00	1.77	5.00	35.00	9.00	1.03	2.00	0.00	1.00	1.03	29.00	30.00	89.00
org.tigris.-scarab.om.-BaseAttributeClassPeer	false	2.00	0.96	407.00	1.55	227.00	1.45	0.00	0.00	0.00	1.34	9.00	38.00	1.00	0.00	1.00	26.00	59.00
org.tigris.-scarab.om.-MITListItemPeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-ModificationPeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-AbstractScarabUser	true	2.00	0.98	1168.00	2.59	861.00	1.98	19.00	88.00	0.00	0.93	4.00	0.00	1.00	0.00	1.00	32.00	228.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-BaseOptionRelationship	false	2.00	0.92	367.00	2.38	263.00	1.97	5.00	28.00	6.00	0.62	2.00	1.00	1.00	0.43	1.00	16.00	69.00
org.tigris.-scarab.om.-BaseDependTypeManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-IssueVoteManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-BaseDependType	false	2.00	0.92	367.00	2.38	263.00	1.97	5.00	28.00	6.00	0.62	2.00	1.00	1.00	0.43	1.00	16.00	69.00
org.tigris.scarab.-om.AttributePeer	false	3.00	1.10	192.00	3.22	138.00	1.78	0.00	0.00	0.00	2.00	9.00	9.00	0.00	0.00	1.00	14.00	29.00
org.tigris.scarab.-om.BaseMITList	false	2.00	0.94	986.00	2.97	767.00	2.18	13.00	60.00	6.00	0.72	2.00	1.00	1.00	0.20	2.00	20.00	181.00
org.tigris.-scarab.om.-BaseAttributeTypeManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-BaseDependTypePeer	false	2.00	0.96	395.00	1.55	219.00	1.45	0.00	0.00	0.00	1.34	7.00	38.00	1.00	0.00	1.00	26.00	59.00
org.tigris.-scarab.om.-BaseROptionOptionManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.-scarab.om.-BaseRModuleOption	false	2.00	0.96	437.00	2.10	294.00	1.48	13.00	39.00	6.00	0.60	2.00	1.00	1.00	0.31	1.00	19.00	84.00
org.tigris.-scarab.om.-BaseScopeManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-BaseActivityPeer	false	2.00	0.94	2506.00	5.82	2230.00	2.27	0.00	0.00	0.00	1.23	21.00	56.00	1.00	0.00	1.00	33.00	326.00
org.tigris.-scarab.om.-BaseAttributeClassManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-BaseIssuePeer	false	2.00	0.93	820.00	2.68	608.00	1.89	0.00	0.00	0.00	1.30	13.00	44.00	1.00	0.00	1.00	31.00	118.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-BaseRModuleIssueTypeManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.-scarab.om.-BaseRModuleIssueTypePeer	false	2.00	0.91	710.00	2.62	506.00	1.75	0.00	0.00	0.00	1.38	15.00	40.00	1.00	0.00	1.00	33.00	105.00
org.tigris.-scarab.om.-UserPreference	true	3.00	0.00	10.00	2.00	2.00	1.00	0.00	1.00	1.00	1.00	0.00	0.00	0.00	3.00	8.00	2.00	2.00
org.tigris.-scarab.om.-OptionRelationshipPeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-ActivitySetManager	false	3.00	0.00	51.00	1.83	20.00	1.17	0.00	1.00	0.00	1.83	0.00	5.00	0.00	0.00	1.00	10.00	11.00
org.tigris.-scarab.om.-TransitionManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-BaseAttachmentTypeManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-BaseGlobalParameterPeer	false	2.00	0.95	448.00	1.64	264.00	1.51	0.00	0.00	0.00	1.33	9.00	39.00	1.00	0.00	1.00	27.00	64.00
org.tigris.-scarab.om.-BaseAttachment	false	2.00	0.96	1124.00	2.78	854.00	1.91	21.00	75.00	6.00	0.66	2.00	1.00	1.00	0.16	1.00	22.00	211.00
org.tigris.-scarab.om.-RModuleAttributeManager	false	3.00	0.00	52.00	1.00	25.00	1.00	0.00	2.00	1.00	1.60	0.00	3.00	0.00	1.50	2.00	12.00	5.00
org.tigris.-scarab.om.-BasePendingGroupUserRolePeer	false	2.00	0.93	533.00	1.72	341.00	1.65	0.00	0.00	0.00	1.42	8.00	40.00	1.00	0.00	1.00	31.00	69.00
org.tigris.scarab.-om.IssueVote	true	3.00	0.00	6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	2.00	0.00
org.tigris.-scarab.om.-BaseDependManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-FrequencyManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.scarab.-om.DependPeer	false	3.00	0.00	9.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00	0.00	1.00	2.00	0.00
org.tigris.-scarab.om.-BaseCondition	false	2.00	0.96	478.00	2.14	328.00	1.47	12.00	42.00	6.00	0.56	2.00	1.00	1.00	0.29	1.00	22.00	92.00
org.tigris.-scarab.om.-BaseAttributeTypePeer	false	2.00	0.95	454.00	1.64	268.00	1.51	0.00	0.00	0.00	1.33	10.00	39.00	1.00	0.00	1.00	27.00	64.00
org.tigris.-scarab.om.-BaseIssueVote	false	2.00	0.96	284.00	1.86	179.00	1.48	8.00	28.00	6.00	0.59	2.00	1.00	1.00	0.43	1.00	19.00	54.00
org.tigris.-scarab.om.-IssueManager	false	3.00	1.00	219.00	3.90	177.00	2.00	0.00	6.00	2.00	0.80	2.00	4.00	0.00	1.00	1.00	20.00	39.00
org.tigris.-scarab.om.-BaseAttributeGroupPeer	false	2.00	0.93	609.00	2.32	413.00	1.68	0.00	0.00	0.00	1.32	13.00	40.00	1.00	0.00	1.00	30.00	93.00
org.tigris.-scarab.om.-AttributeOptionManager	false	3.00	0.00	25.00	1.00	9.00	1.00	0.00	3.00	2.00	0.33	0.00	0.00	0.00	2.00	1.00	5.00	3.00
org.tigris.scarab.-om.MITListPeer	false	3.00	0.00	11.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.00	0.00	0.00	0.00	1.00	2.00	0.00
org.tigris.scarab.-om.Query	true	3.00	0.97	277.00	2.29	202.00	1.88	1.00	16.00	5.00	0.94	1.00	1.00	0.00	0.94	22.00	20.00	39.00
org.tigris.-scarab.om.-AttributeGroupPeer	false	3.00	0.00	25.00	2.00	12.00	2.00	0.00	0.00	0.00	1.00	2.00	1.00	0.00	0.00	1.00	5.00	2.00
org.tigris.-scarab.om.-BaseUserVotePeer	false	2.00	0.94	530.00	1.72	338.00	1.65	0.00	0.00	0.00	1.38	9.00	40.00	1.00	0.00	1.00	31.00	69.00
org.tigris.-scarab.om.-BaseIssueTemplateInfoPeer	false	2.00	0.93	541.00	1.98	351.00	1.68	0.00	0.00	0.00	1.32	10.00	40.00	1.00	0.00	1.00	30.00	79.00
org.tigris.-scarab.om.-BaseAttributeGroupManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-BaseROptionOptionPeer	false	2.00	0.93	719.00	2.09	509.00	1.86	0.00	0.00	0.00	1.34	10.00	44.00	1.00	0.00	1.00	31.00	92.00
org.tigris.-scarab.om.-BaseScarabUserManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	14.00	22.00
org.tigris.-scarab.om.-BaseOptionRelationshipManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-ScarabModuleContainer	true	5.00	0.00	16.00	1.00	3.00	1.00	0.00	3.00	2.00	0.00	0.00	0.00	0.00	3.33	0.00	1.00	3.00
org.tigris.-scarab.om.-RIssueTypeAttribute	true	3.00	0.00	111.00	2.00	89.00	1.75	0.00	4.00	0.00	0.25	0.00	0.00	0.00	0.00	16.00	9.00	8.00
org.tigris.-scarab.om.-BaseQueryManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-IssueTemplateInfoPeer	false	3.00	0.93	177.00	3.40	138.00	2.40	0.00	0.00	0.00	3.00	7.00	5.00	0.00	0.00	1.00	13.00	17.00
org.tigris.-scarab.om.-ModificationManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-BaseRModuleAttribute	false	2.00	0.96	778.00	2.58	581.00	1.80	17.00	54.00	6.00	0.65	2.00	1.00	1.00	0.22	1.00	21.00	142.00
org.tigris.-scarab.om.-BaseIssueTemplateInfoManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-BaseGlobalParameterManager	true	3.00	0.00	6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	2.00	0.00
org.tigris.-scarab.om.-AttributeClass	false	3.00	0.96	122.00	3.00	82.00	2.00	0.00	0.00	0.00	1.43	4.00	7.00	0.00	0.00	1.00	9.00	21.00
org.tigris.-scarab.om.-IssueTypePeer																		

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-om.Attribute	true	3.00	0.99	656.00	1.51	509.00	1.42	6.00	36.00	3.00	0.42	12.00	7.00	0.00	0.25	87.00	30.00	65.00
org.tigris.-scarab.om.-RIssueTypeOption	true	3.00	0.00	55.00	3.00	41.00	2.50	0.00	2.00	0.00	1.00	0.00	0.00	0.00	0.00	11.00	9.00	6.00
org.tigris.-scarab.om.-RModuleIssueTypeManager	false	3.00	0.00	47.00	1.50	26.00	1.25	0.00	2.00	1.00	0.75	0.00	2.00	0.00	1.50	1.00	12.00	6.00
org.tigris.-scarab.om.-MITListManager	false	3.00	0.00	121.00	2.00	84.00	1.57	0.00	1.00	0.00	1.71	0.00	6.00	0.00	0.00	2.00	12.00	14.00
org.tigris.-scarab.om.-BaseModificationManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.-scarab.om.-RQueryUserManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-RAttributeAttributeGroupPeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-AttributeOption	true	3.00	0.98	313.00	1.40	234.00	1.30	5.00	24.00	3.00	0.40	3.00	6.00	0.00	0.38	54.00	19.00	42.00
org.tigris.-scarab.om.-RIssueTypeOptionManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-BaseConditionManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-BaseRModuleOptionManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.-scarab.om.-RModuleOptionManager	false	3.00	0.00	42.00	1.00	19.00	1.00	0.00	2.00	1.00	1.75	0.00	2.00	0.00	1.50	1.00	12.00	4.00
org.tigris.-scarab.om.-BaseAttributeValuePeer	false	2.00	0.92	1042.00	3.17	818.00	1.98	0.00	0.00	0.00	1.30	13.00	47.00	1.00	0.00	1.00	32.00	149.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-om.BaseAttribute	false	2.00	0.97	2356.00	3.27	1882.00	2.30	32.00	132.00	6.00	0.77	2.00	1.00	1.00	0.09	1.00	27.00	435.00
org.tigris.-scarab.om.-BaseFrequencyManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.scarab.-om.BaseDepend	false	2.00	0.94	700.00	2.58	521.00	1.92	11.00	49.00	6.00	0.66	2.00	1.00	1.00	0.24	1.00	19.00	129.00
org.tigris.-scarab.om.-RModuleUserAttribute	true	3.00	0.00	40.00	5.00	31.00	4.00	0.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	21.00	8.00	5.00
org.tigris.-scarab.om.-ScarabUserImplPeer	false	5.00	0.00	11.00	1.00	3.00	1.00	0.00	0.00	0.00	1.00	0.00	1.00	0.00	0.00	1.00	4.00	1.00
org.tigris.-scarab.om.-ReportManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.scarab.-om.Modification	true	3.00	0.00	6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.00	2.00	0.00
org.tigris.-scarab.om.-ROptionOptionPeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-BaseRModuleUserAttribute	false	2.00	0.96	477.00	2.11	320.00	1.47	13.00	44.00	6.00	0.56	2.00	1.00	1.00	0.27	1.00	20.00	95.00
org.tigris.-scarab.om.-BaseRIssueTypeOptionManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.-scarab.om.-QueryManager	false	3.00	0.00	27.00	1.00	11.00	1.00	0.00	3.00	2.00	0.33	0.00	0.00	0.00	2.00	1.00	3.00	3.00
org.tigris.-scarab.om.-BasePendingGroupUserRole	false	2.00	0.95	288.00	1.86	183.00	1.48	8.00	28.00	6.00	0.62	2.00	1.00	1.00	0.43	1.00	18.00	54.00
org.tigris.-scarab.om.-PendingGroupUserRoleManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-BaseAttachmentManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.om.ScopePeer	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	1.00	3.00	1.00
org.tigris.scarab.om.ParentChildAttributeOption	true	1.00	0.96	243.00	1.32	137.00	1.29	8.00	23.00	1.00	0.54	2.00	5.00	0.00	0.04	5.00	18.00	37.00
org.tigris.scarab.om.AttributeGroupManager	false	3.00	0.00	67.00	1.57	38.00	1.43	0.00	7.00	3.00	0.43	0.00	0.00	0.00	1.29	1.00	12.00	11.00
org.tigris.scarab.om.BaseActivitySetTypeManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.scarab.om.IssueTemplateInfo	false	3.00	0.00	89.00	2.40	63.00	2.00	0.00	4.00	0.00	1.40	0.00	1.00	0.00	0.00	12.00	11.00	12.00
org.tigris.scarab.om.UserVoteManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.scarab.om.MITList	true	3.00	0.97	921.00	2.53	696.00	2.02	4.00	53.00	6.00	0.66	0.00	0.00	0.00	0.34	36.00	32.00	134.00
org.tigris.scarab.om.MITList$ItemsIterator	true	1.00	0.33	43.00	1.50	25.00	1.25	2.00	4.00	0.00	0.25	0.00	0.00	0.00	0.00	2.00	32.00	6.00
org.tigris.scarab.om.BaseReportPeer	false	2.00	0.93	1039.00	3.20	815.00	1.98	0.00	0.00	0.00	1.28	15.00	46.00	1.00	0.00	1.00	32.00	147.00
org.tigris.scarab.om.BaseIssueTemplateInfo	false	2.00	0.95	330.00	2.03	217.00	1.50	8.00	31.00	6.00	0.53	2.00	1.00	1.00	0.39	1.00	17.00	65.00
org.tigris.scarab.om.BaseAttributePeer	false	2.00	0.93	832.00	2.68	616.00	1.89	0.00	0.00	0.00	1.30	15.00	44.00	1.00	0.00	1.00	31.00	118.00
org.tigris.scarab.om.BaseIssueManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.scarab.om.RIssueTypeAttributePeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.scarab.om.ScarabUser	false	0.00	1.01	114.00	1.00	109.00	1.00	0.00	72.00	0.00	0.83	1.00	0.00	1.00	0.00	115.00	17.00	72.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-AttributeTypePeer	false	3.00	0.00	8.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00	0.00	1.00	2.00	0.00
org.tigris.-scarab.om.-UserPreferenceManager	false	3.00	0.00	33.00	2.50	20.00	1.50	0.00	1.00	0.00	0.50	0.00	1.00	0.00	0.00	1.00	5.00	5.00
org.tigris.-scarab.om.-BaseAttributeValue	false	2.00	0.96	477.00	2.16	323.00	1.48	13.00	43.00	6.00	0.57	2.00	1.00	10.00	0.28	1.00	19.00	95.00
org.tigris.-scarab.om.-BaseRQueryUserPeer	false	2.00	0.89	923.00	3.14	713.00	1.95	0.00	0.00	0.00	1.34	10.00	44.00	1.00	0.00	1.00	35.00	138.00
org.tigris.-scarab.om.-ScarabModulePeer	false	3.00	0.00	26.00	1.50	19.00	1.50	0.00	0.00	0.00	0.50	0.00	2.00	0.00	0.00	1.00	7.00	3.00
org.tigris.-scarab.om.-BaseModificationPeer	false	2.00	0.95	464.00	1.66	276.00	1.53	0.00	0.00	0.00	1.39	11.00	38.00	1.00	0.00	1.00	29.00	63.00
org.tigris.-scarab.om.-BaseIssueTypePeer	false	2.00	0.92	559.00	2.39	371.00	1.55	0.00	0.00	0.00	1.34	13.00	38.00	1.00	0.00	1.00	28.00	91.00
org.tigris.scarab.-om.ActivitySet	true	3.00	0.88	108.00	1.78	63.00	1.22	0.00	9.00	0.00	1.67	1.00	0.00	0.00	0.00	33.00	13.00	16.00
org.tigris.-scarab.om.-FrequencyPeer	false	3.00	0.00	9.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	1.00	3.00	1.00
org.tigris.-scarab.om.-AttributeManager	false	3.00	0.00	25.00	1.00	9.00	1.00	0.00	3.00	2.00	0.33	0.00	0.00	0.00	2.00	1.00	5.00	3.00
org.tigris.-scarab.om.-ModuleManager	false	3.00	0.00	163.00	3.00	122.00	1.78	0.00	7.00	2.00	1.00	0.00	2.00	0.00	0.86	2.00	22.00	27.00
org.tigris.-scarab.om.-BaseRAttributeAttributeGroup	false	2.00	0.96	284.00	1.86	179.00	1.48	8.00	28.00	6.00	0.59	2.00	1.00	1.00	0.43	1.00	18.00	54.00
org.tigris.scarab.-om.AttributeType	true	3.00	1.00	53.00	2.00	39.00	2.00	0.00	1.00	1.00	0.50	3.00	1.00	0.00	3.00	11.00	9.00	4.00
org.tigris.-scarab.om.-BaseRModuleOptionPeer	false	2.00	0.91	864.00	2.68	648.00	1.95	0.00	0.00	0.00	1.39	12.00	44.00	1.00	0.00	1.00	34.00	118.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-IssueTemplateInfoManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-DependManager	false	3.00	0.00	32.00	1.00	19.00	1.00	0.00	2.00	1.00	0.50	0.00	0.00	0.00	1.50	1.00	4.00	2.00
org.tigris.-scarab.om.-BasePendingGroupUserRoleManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.-scarab.om.-BaseRModuleUserAttributeManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-BaseIssueTypeManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-BaseRQueryUser	false	2.00	0.96	373.00	2.00	244.00	1.47	11.00	35.00	6.00	0.58	2.00	1.00	1.00	0.34	1.00	20.00	72.00
org.tigris.-scarab.om.-BaseUserPreference	false	2.00	0.95	301.00	2.03	198.00	1.52	7.00	28.00	6.00	0.52	2.00	1.00	1.00	0.43	1.00	16.00	59.00
org.tigris.-scarab.om.-WorkflowRules	true	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00
org.tigris.-scarab.om.-BaseUserPreferenceManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-RModuleIssueTypePeer	false	3.00	0.00	14.00	1.00	3.00	1.00	0.00	0.00	0.00	1.00	1.00	1.00	0.00	0.00	1.00	4.00	1.00
org.tigris.-scarab.om.-BaseTransitionPeer	false	2.00	0.92	764.00	2.55	556.00	1.89	0.00	0.00	0.00	1.30	11.00	44.00	1.00	0.00	1.00	30.00	112.00
org.tigris.scarab.-om.UserVote	true	3.00	0.00	6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	2.00	0.00
org.tigris.-scarab.om.-ScarabUserImpl	false	5.00	0.90	676.00	1.41	375.00	1.23	1.00	79.00	2.00	0.84	5.00	2.00	0.00	0.13	19.00	31.00	114.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-AttributeValue	true	3.00	0.95	653.00	2.78	485.00	1.91	12.00	42.00	12.00	0.58	1.00	3.00	9.00	0.86	37.00	25.00	125.00
org.tigris.scarab.-om.MITListItem	true	3.00	0.75	110.00	2.33	75.00	1.78	0.00	9.00	4.00	0.22	1.00	0.00	0.00	1.33	18.00	10.00	21.00
org.tigris.scarab.-om.QueryPeer	false	3.00	0.99	210.00	2.33	151.00	1.67	0.00	0.00	0.00	3.00	12.00	9.00	0.00	0.00	1.00	11.00	21.00
org.tigris.-scarab.om.-BaseRIssueTypeOption	false	2.00	0.96	374.00	2.06	248.00	1.49	11.00	34.00	6.00	0.57	2.00	1.00	1.00	0.35	1.00	18.00	72.00
org.tigris.-scarab.om.-BaseActivitySetManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-ConditionPeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-BaseRIssueTypeAttributeManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.-scarab.om.-BaseRModuleIssueType	false	2.00	0.96	753.00	2.59	559.00	1.80	17.00	53.00	6.00	0.63	2.00	1.00	1.00	0.23	1.00	20.00	140.00
org.tigris.-scarab.om.-BaseRModuleUserAttributePeer	false	2.00	0.94	1160.00	3.21	934.00	1.98	0.00	0.00	0.00	1.27	12.00	48.00	1.00	0.00	1.00	31.00	154.00
org.tigris.-scarab.om.-BaseIssueVotePeer	false	2.00	0.94	524.00	1.72	334.00	1.65	0.00	0.00	0.00	1.38	8.00	40.00	1.00	0.00	1.00	32.00	69.00
org.tigris.-scarab.om.-AttributeValueManager	false	3.00	0.00	21.00	1.00	8.00	1.00	0.00	2.00	1.00	0.50	0.00	0.00	0.00	1.50	1.00	4.00	2.00
org.tigris.-scarab.om.-DependTypeManager	false	3.00	0.92	71.00	1.60	40.00	1.40	0.00	3.00	1.00	0.40	3.00	2.00	0.00	1.00	1.00	8.00	8.00
org.tigris.-scarab.om.-ActivitySetType	true	3.00	0.00	11.00	1.00	1.00	1.00	0.00	0.00	0.00	1.00	0.00	1.00	0.00	0.00	9.00	4.00	1.00
org.tigris.-scarab.om.-DependTypePeer	false	3.00	0.00	8.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.00	0.00	0.00	0.00	1.00	2.00	0.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-ActivityManager	false	3.00	0.00	281.00	1.44	136.00	1.22	0.00	2.00	1.00	6.06	0.00	16.00	0.00	1.50	1.00	16.00	26.00
org.tigris.-scarab.om.-BaseTransition	false	2.00	0.95	626.00	2.48	456.00	1.81	12.00	47.00	6.00	0.62	2.00	1.00	1.00	0.26	1.00	18.00	119.00
org.tigris.scarab.-om.Attachment	true	3.00	1.02	292.00	2.47	228.00	1.76	1.00	13.00	3.00	0.65	6.00	4.00	0.00	0.69	38.00	24.00	42.00
org.tigris.-scarab.om.-AttributeValuePeer	false	3.00	1.00	47.00	1.67	35.00	1.67	0.00	0.00	0.00	1.33	3.00	3.00	0.00	0.00	1.00	11.00	5.00
org.tigris.-scarab.om.-BaseAttributeOptionPeer	false	2.00	0.94	488.00	1.90	304.00	1.62	0.00	0.00	0.00	1.33	9.00	39.00	1.00	0.00	1.00	29.00	74.00
org.tigris.-scarab.om.-BaseUserVoteManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.-scarab.om.-IssueVotePeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-BaseScarabModule	false	3.00	0.98	2818.00	3.31	2255.00	2.30	41.00	158.00	7.00	0.78	2.00	1.00	2.00	0.13	1.00	29.00	527.00
org.tigris.scarab.-om.IssuePeer	false	3.00	1.00	44.00	1.33	29.00	1.33	0.00	0.00	0.00	1.00	4.00	3.00	0.00	0.00	1.00	7.00	4.00
org.tigris.scarab.-om.RQueryUser	true	3.00	0.00	16.00	2.00	7.00	2.00	0.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	9.00	4.00	2.00
org.tigris.-scarab.om.-BaseAttachmentType	false	2.00	0.92	423.00	2.47	308.00	1.97	6.00	31.00	6.00	0.62	2.00	1.00	1.00	0.39	1.00	16.00	79.00
org.tigris.-scarab.om.-TransitionPeer	false	3.00	0.00	109.00	2.50	92.00	1.75	0.00	0.00	0.00	1.25	0.00	4.00	0.00	0.00	2.00	10.00	10.00
org.tigris.-scarab.om.-BaseMITListItemManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-BaseUserPreferencePeer	false	2.00	0.95	452.00	1.59	266.00	1.51	0.00	0.00	0.00	1.33	10.00	39.00	1.00	0.00	1.00	27.00	62.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-AttachmentPeer	false	3.00	0.00	25.00	2.00	12.00	2.00	0.00	0.00	0.00	1.00	2.00	1.00	0.00	0.00	1.00	5.00	2.00
org.tigris.-scarab.om.-BaseMITListItem	false	2.00	0.95	329.00	1.97	213.00	1.48	8.00	32.00	6.00	0.55	2.00	1.00	1.00	0.38	1.00	18.00	65.00
org.tigris.-scarab.om.-BaseAttributeValueManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-RModuleUserAttributePeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.scarab.-om.BaseActivity	false	2.00	0.98	863.00	2.31	603.00	1.45	26.00	74.00	6.00	0.56	2.00	1.00	1.00	0.16	1.00	23.00	173.00
org.tigris.-scarab.om.-RModuleOptionPeer	false	3.00	0.00	14.00	1.00	3.00	1.00	0.00	0.00	0.00	1.00	1.00	1.00	0.00	0.00	1.00	4.00	1.00
org.tigris.-scarab.om.-RIssueTypeAttributeManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-BaseAttributeOption	false	2.00	0.97	2514.00	3.43	2034.00	2.47	28.00	133.00	6.00	0.82	2.00	1.00	1.00	0.09	1.00	23.00	459.00
org.tigris.scarab.-om.BaseQuery	false	2.00	0.97	931.00	2.51	676.00	1.64	24.00	72.00	6.00	0.59	2.00	1.00	1.00	0.17	1.00	24.00	183.00
org.tigris.-scarab.om.-GlobalParameterPeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-AttachmentTypePeer	false	3.00	0.00	12.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	0.00	0.00	0.00	1.00	3.00	0.00
org.tigris.-scarab.om.-GlobalParameter	true	3.00	0.00	13.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	0.00	0.00	0.00	6.00	3.00	0.00
org.tigris.-scarab.om.-ScarabBaseObject	true	2.00	0.76	39.00	1.00	8.00	1.00	3.00	8.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	2.00	8.00
org.tigris.-scarab.om.-BaseAttachmentTypePeer	false	2.00	0.95	433.00	1.76	255.00	1.55	0.00	0.00	0.00	1.34	8.00	38.00	1.00	0.00	1.00	28.00	67.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-BaseMITListItemPeer	false	2.00	0.94	716.00	2.23	512.00	1.80	0.00	0.00	0.00	1.30	9.00	44.00	1.00	0.00	1.00	29.00	98.00
org.tigris.-scarab.om.-AttributeGroup	true	3.00	0.94	488.00	2.44	406.00	1.72	0.00	17.00	2.00	0.50	2.00	1.00	0.00	0.35	25.00	29.00	44.00
org.tigris.-scarab.om.-BaseConditionPeer	false	2.00	0.93	1160.00	3.21	936.00	1.98	0.00	0.00	0.00	1.27	11.00	48.00	1.00	0.00	1.00	31.00	154.00
org.tigris.-scarab.om.-BaseMITListManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-ConditionManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-AbstractScarabModule	true	2.00	1.00	1647.00	2.75	1301.00	2.02	3.00	85.00	1.00	1.02	17.00	0.00	3.00	0.02	1.00	49.00	234.00
org.tigris.-scarab.om.-BaseFrequencyPeer	false	2.00	0.96	393.00	1.50	217.00	1.45	0.00	0.00	0.00	1.34	7.00	38.00	1.00	0.00	1.00	26.00	57.00
org.tigris.-scarab.om.-ActivitySetTypePeer	false	3.00	0.00	9.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	0.00	0.00	0.00	1.00	2.00	0.00
org.tigris.-scarab.om.-BaseRIssueTypeAttribute	false	2.00	0.96	434.00	2.15	294.00	1.49	13.00	38.00	6.00	0.56	2.00	1.00	1.00	0.32	1.00	18.00	84.00
org.tigris.-scarab.om.-BaseRAttributeAttributeGroupManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.scarab.-om.IssueType	true	3.00	0.99	869.00	2.61	634.00	2.20	2.00	45.00	2.00	0.91	15.00	1.00	0.00	0.13	88.00	27.00	120.00
org.tigris.-scarab.om.-ActivitySetPeer	false	3.00	0.00	25.00	2.00	12.00	2.00	0.00	0.00	0.00	1.00	2.00	1.00	0.00	0.00	1.00	5.00	2.00
org.tigris.-scarab.om.-AttachmentManager	false	3.00	0.00	53.00	1.00	19.00	1.00	0.00	2.00	1.00	2.33	0.00	4.00	0.00	1.50	1.00	10.00	6.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-RModuleIssueType	true	3.00	0.00	246.00	2.40	191.00	1.87	0.00	15.00	7.00	0.47	0.00	0.00	0.00	1.40	36.00	25.00	36.00
org.tigris.-scarab.om.-AttributeClassManager	false	3.00	0.00	26.00	1.00	10.00	1.00	0.00	3.00	2.00	0.33	0.00	0.00	0.00	2.00	1.00	5.00	3.00
org.tigris.-scarab.om.-BaseGlobalParameter	false	2.00	0.94	271.00	1.96	175.00	1.52	6.00	26.00	6.00	0.52	2.00	1.00	1.00	0.46	1.00	15.00	53.00
org.tigris.-scarab.om.-BaseActivitySet	false	2.00	0.95	885.00	2.76	674.00	2.03	13.00	58.00	6.00	0.69	2.00	1.00	1.00	0.21	1.00	22.00	163.00
org.tigris.-scarab.om.-AttributeOptionPeer	false	3.00	0.00	21.00	2.00	14.00	2.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	1.00	4.00	2.00
org.tigris.scarab.-om.Depend	true	3.00	0.90	98.00	1.55	50.00	1.27	3.00	10.00	0.00	0.64	0.00	1.00	0.00	0.00	16.00	10.00	17.00
org.tigris.scarab.-om.DependType	true	3.00	0.00	6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	2.00	0.00
org.tigris.-scarab.om.-BaseAttributeClass	false	2.00	0.94	375.00	2.35	265.00	1.84	7.00	30.00	6.00	0.58	2.00	1.00	1.00	0.40	1.00	16.00	73.00
org.tigris.scarab.-om.BaseScope	false	2.00	0.93	815.00	2.98	635.00	2.28	9.00	49.00	6.00	0.74	2.00	1.00	1.00	0.24	1.00	18.00	149.00
org.tigris.-scarab.om.-RModuleAttributePeer	false	3.00	1.00	43.00	1.33	21.00	1.33	0.00	0.00	0.00	1.00	3.00	3.00	0.00	0.00	1.00	7.00	4.00
org.tigris.-scarab.om.-BaseModuleManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	14.00	22.00
org.tigris.-scarab.om.-BaseRQueryUserManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.-scarab.om.-BaseModification	false	2.00	0.95	315.00	2.07	210.00	1.52	9.00	28.00	6.00	0.55	2.00	1.00	1.00	0.43	1.00	17.00	60.00
org.tigris.scarab.-om.ReportPeer	false	3.00	1.00	49.00	2.00	28.00	1.67	0.00	0.00	0.00	1.00	2.00	3.00	0.00	0.00	1.00	8.00	6.00
org.tigris.-scarab.om.-AttachmentTypeManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-ActivitySetTypeManager	false	3.00	1.00	47.00	2.50	30.00	2.00	0.00	1.00	0.00	0.50	2.00	1.00	0.00	0.00	1.00	7.00	5.00
org.tigris.-scarab.om.-AttachmentType	true	3.00	0.00	17.00	2.00	8.00	2.00	0.00	0.00	0.00	1.00	0.00	1.00	0.00	0.00	7.00	7.00	2.00
org.tigris.-scarab.om.-BaseUserVote	false	2.00	0.96	314.00	1.94	202.00	1.48	9.00	30.00	6.00	0.58	2.00	1.00	1.00	0.40	1.00	18.00	60.00
org.tigris.scarab.-om.Transition	true	3.00	0.00	194.00	2.36	146.00	1.93	0.00	14.00	1.00	0.29	0.00	0.00	0.00	0.21	13.00	16.00	33.00
org.tigris.-scarab.om.-ROptionOption	true	3.00	0.95	104.00	1.25	60.00	1.25	1.00	6.00	1.00	0.67	1.00	6.00	0.00	0.50	12.00	15.00	15.00
org.tigris.-scarab.om.-RModuleOption	true	3.00	0.95	184.00	1.92	117.00	1.75	1.00	11.00	4.00	0.50	1.00	1.00	0.00	1.09	26.00	23.00	23.00
org.tigris.-scarab.om.-BaseAttributeGroup	false	2.00	0.96	579.00	2.46	417.00	1.74	13.00	45.00	6.00	0.59	2.00	1.00	1.00	0.27	1.00	18.00	113.00
org.tigris.-scarab.om.-GlobalParameterManager	false	3.00	0.81	290.00	3.50	227.00	2.50	0.00	2.00	1.00	1.86	4.00	12.00	0.00	1.50	2.00	14.00	49.00
org.tigris.-scarab.om.-BaseROptionOption	false	2.00	0.96	373.00	2.00	244.00	1.47	11.00	35.00	6.00	0.58	2.00	1.00	1.00	0.34	1.00	18.00	72.00
org.tigris.scarab.-om.Report	true	3.00	0.00	25.00	1.33	8.00	1.33	0.00	3.00	3.00	0.33	0.00	0.00	0.00	3.00	13.00	4.00	4.00
org.tigris.-scarab.om.-BaseReportManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-OptionWorkflow	true	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00
org.tigris.scarab.-om.UserVotePeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-BaseAttributeOptionManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-om.Scope	true	3.00	0.00	8.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00	0.00	11.00	3.00	0.00
org.tigris.-scarab.om.-BaseQueryPeer	false	2.00	0.90	1871.00	5.74	1621.00	2.14	0.00	0.00	0.00	1.26	20.00	50.00	1.00	0.00	1.00	34.00	287.00
org.tigris.scarab.-om.Activity	true	3.00	1.00	120.00	3.00	90.00	2.33	2.00	6.00	3.00	0.50	1.00	0.00	0.00	1.50	26.00	14.00	18.00
org.tigris.-scarab.om.-BaseAttributeType	false	2.00	0.94	486.00	2.45	351.00	1.84	9.00	37.00	6.00	0.61	2.00	1.00	1.00	0.32	1.00	17.00	93.00
org.tigris.scarab.-om.ActivityPeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-BaseOptionRelationshipPeer	false	2.00	0.96	395.00	1.55	219.00	1.45	0.00	0.00	0.00	1.34	7.00	38.00	1.00	0.00	1.00	26.00	59.00
org.tigris.scarab.-om.BaseReport	false	2.00	0.96	538.00	2.23	370.00	1.48	15.00	47.00	6.00	0.54	2.00	1.00	1.00	0.26	1.00	19.00	107.00
org.tigris.-scarab.om.-BaseAttachmentPeer	false	2.00	0.93	1007.00	3.07	779.00	1.98	0.00	0.00	0.00	1.28	17.00	46.00	1.00	0.00	1.00	31.00	141.00
org.tigris.-scarab.om.-RModuleAttribute	true	3.00	1.00	314.00	2.56	249.00	2.31	0.00	15.00	5.00	0.56	2.00	1.00	0.00	1.00	35.00	23.00	41.00
org.tigris.-scarab.om.-BaseRAttributeAttributeGroupPeer	false	2.00	0.94	524.00	1.72	334.00	1.65	0.00	0.00	0.00	1.38	8.00	40.00	1.00	0.00	1.00	31.00	69.00
org.tigris.-scarab.om.-BaseRIssueTypeOptionPeer	false	2.00	0.91	638.00	2.32	442.00	1.75	0.00	0.00	0.00	1.38	11.00	40.00	1.00	0.00	1.00	33.00	93.00
org.tigris.-scarab.om.-PendingGroupUserRole	true	3.00	0.00	15.00	1.00	5.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	8.00	3.00	1.00
org.tigris.-scarab.om.-BaseIssueVoteManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.-scarab.om.-PendingGroupUserRolePeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-BaseRIssueTypeAttributePeer	false	2.00	0.88	794.00	3.22	594.00	1.75	0.00	0.00	0.00	1.38	13.00	40.00	1.00	0.00	1.00	33.00	129.00
org.tigris.-scarab.om.-RQueryUserPeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-BaseMITListPeer	false	2.00	0.93	534.00	2.15	348.00	1.62	0.00	0.00	0.00	1.33	10.00	39.00	1.00	0.00	1.00	29.00	84.00
org.tigris.-scarab.om.-BaseIssueType	false	2.00	0.97	2478.00	3.38	1994.00	2.39	32.00	135.00	6.00	0.79	2.00	1.00	1.00	0.09	1.00	26.00	459.00
org.tigris.-scarab.om.-OptionRelationship	true	3.00	0.00	7.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	5.00	3.00	0.00
org.tigris.-scarab.om.-BaseScarabModulePeer	false	2.00	0.94	817.00	2.73	589.00	1.82	0.00	0.00	0.00	1.32	21.00	44.00	1.00	0.00	1.00	29.00	120.00
org.tigris.-scarab.om.-RAttributeAttributeGroup	true	3.00	0.00	13.00	1.00	4.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	12.00	3.00	1.00
org.tigris.-scarab.om.-BaseRModuleAttributeManager	false	2.00	1.00	118.00	1.00	33.00	1.00	0.00	9.00	0.00	0.90	2.00	11.00	1.00	0.00	1.00	12.00	20.00
org.tigris.-scarab.om.-BaseScarabUserImplPeer	false	4.00	1.08	118.00	1.78	75.00	1.78	0.00	0.00	0.00	1.22	3.00	9.00	1.00	0.00	1.00	14.00	16.00
org.tigris.scarab.-om.Issue	true	3.00	1.00	3036.00	2.88	2350.00	1.95	1.00	128.00	7.00	1.41	34.00	5.00	1.00	0.16	78.00	53.00	383.00
org.tigris.-scarab.om.-Issue$FederatedId	true	1.00	0.57	98.00	1.91	57.00	1.55	3.00	11.00	2.00	0.82	0.00	0.00	0.00	0.18	7.00	53.00	21.00
org.tigris.-scarab.om.-BaseActivitySetPeer	false	2.00	0.94	722.00	2.23	516.00	1.80	0.00	0.00	0.00	1.30	10.00	44.00	1.00	0.00	1.00	29.00	98.00
org.tigris.-scarab.om.-BaseAttributeManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-BaseActivitySetType	false	2.00	0.92	367.00	2.38	263.00	1.97	5.00	28.00	6.00	0.62	2.00	1.00	1.00	0.43	1.00	16.00	69.00
org.tigris.-scarab.om.-AttributeClassPeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.scarab.-om.Module	false	0.00	1.01	170.00	1.00	161.00	1.00	4.00	98.00	0.00	0.93	0.00	0.00	6.00	0.00	104.00	20.00	98.00
org.tigris.-scarab.om.-BaseRModuleAttributePeer	false	2.00	0.87	1116.00	4.05	896.00	1.95	0.00	0.00	0.00	1.39	14.00	44.00	1.00	0.00	1.00	34.00	178.00
org.tigris.-scarab.om.-BaseScopePeer	false	2.00	0.96	393.00	1.50	217.00	1.45	0.00	0.00	0.00	1.34	7.00	38.00	1.00	0.00	1.00	26.00	57.00
org.tigris.scarab.-om.BaseIssue	false	2.00	0.97	1848.00	3.18	1464.00	2.25	26.00	107.00	6.00	0.76	2.00	1.00	2.00	0.11	2.00	25.00	343.00
org.tigris.scarab.-om.Condition	true	3.00	0.00	40.00	9.00	28.00	3.00	0.00	2.00	1.00	1.00	0.00	0.00	0.00	1.50	18.00	5.00	18.00
org.tigris.-scarab.om.-MITListItemManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.om.-UserPreferencePeer	false	3.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.-scarab.om.-RAttributeAttributeGroupManager	false	3.00	0.00	25.00	1.00	9.00	1.00	0.00	3.00	2.00	0.33	0.00	0.00	0.00	2.00	1.00	5.00	3.00
org.tigris.-scarab.om.-AttributeTypeManager	false	3.00	0.00	26.00	1.00	10.00	1.00	0.00	3.00	2.00	0.33	0.00	0.00	0.00	2.00	1.00	5.00	3.00
org.tigris.-scarab.om.-ScarabUserManager	false	3.00	0.00	56.00	2.00	32.00	1.60	0.00	3.00	1.00	1.20	0.00	2.00	0.00	1.00	1.00	8.00	10.00
org.tigris.-scarab.om.-BaseActivityManager	false	2.00	1.00	128.00	1.00	35.00	1.00	0.00	9.00	0.00	0.95	2.00	13.00	1.00	0.00	1.00	13.00	22.00
org.tigris.-scarab.om.-ScopeManager	false	3.00	0.00	10.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-om.Conditioned	false	0.00	0.00	10.00	1.00	6.00	1.00	0.00	4.00	0.00	0.50	0.00	0.00	4.00	0.00	4.00	1.00	4.00
org.tigris.-scarab.om.map.-AttributeTypeMapBuilder	false	1.00	0.75	30.00	1.00	14.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-ROptionOptionMapBuilder	false	1.00	0.75	33.00	1.00	17.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-MITListItemMapBuilder	false	1.00	0.75	33.00	1.00	17.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.om.map.-RModuleOptionMapBuilder	false	1.00	0.75	35.00	1.00	19.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-AttributeGroupMapBuilder	false	1.00	0.75	35.00	1.00	19.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-UserVoteMapBuilder	false	1.00	0.75	30.00	1.00	14.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-RIssueTypeOptionMapBuilder	false	1.00	0.75	32.00	1.00	16.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-ActivitySetMapBuilder	false	1.00	0.75	34.00	1.00	18.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	7.00	3.00
org.tigris.-scarab.om.map.-AttachmentMapBuilder	false	1.00	0.75	43.00	1.00	27.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	7.00	3.00
org.tigris.-scarab.om.map.-ConditionMapBuilder	false	1.00	0.75	39.00	1.00	23.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.om.map.-RModuleUserAttributeMapBuilder	false	1.00	0.75	40.00	1.00	24.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.om.map.-IssueTemplateInfoMapBuilder	false	1.00	0.75	31.00	1.00	15.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.map.-RIssueTypeAttributeMapBuilder	false	1.00	0.75	34.00	1.00	18.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-AttributeClassMapBuilder	false	1.00	0.75	27.00	1.00	11.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-AttachmentTypeMapBuilder	false	1.00	0.75	26.00	1.00	10.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-ScarabModuleMapBuilder	false	1.00	0.75	42.00	1.00	26.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-FrequencyMapBuilder	false	1.00	0.75	24.00	1.00	8.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-QueryMapBuilder	false	1.00	0.75	50.00	1.00	34.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	7.00	3.00
org.tigris.-scarab.om.map.-UserPreferenceMapBuilder	false	1.00	0.75	29.00	1.00	13.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.om.map.-IssueTypeMapBuilder	false	1.00	0.75	33.00	1.00	17.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-OptionRelationshipMapBuilder	false	1.00	0.75	25.00	1.00	9.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-DependMapBuilder	false	1.00	0.75	34.00	1.00	18.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.om.map.-IssueVoteMapBuilder	false	1.00	0.75	29.00	1.00	13.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.om.map.-RModuleAttributeMapBuilder	false	1.00	0.75	37.00	1.00	21.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-IssueMapBuilder	false	1.00	0.75	37.00	1.00	21.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.map.-PendingGroupUserRoleMapBuilder	false	1.00	0.75	29.00	1.00	13.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-RQueryUserMapBuilder	false	1.00	0.75	33.00	1.00	17.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.om.map.-ActivityMapBuilder	false	1.00	0.75	57.00	1.00	41.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	7.00	3.00
org.tigris.-scarab.om.map.-AttributeOptionMapBuilder	false	1.00	0.75	29.00	1.00	13.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-ModificationMapBuilder	false	1.00	0.75	29.00	1.00	13.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.om.map.-ReportMapBuilder	false	1.00	0.75	41.00	1.00	25.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.om.map.-ActivitySetTypeMapBuilder	false	1.00	0.75	25.00	1.00	9.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-RModuleIssueTypeMapBuilder	false	1.00	0.75	36.00	1.00	20.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-GlobalParameterMapBuilder	false	1.00	0.75	29.00	1.00	13.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-RAttributeAttributeGroupMapBuilder	false	1.00	0.75	30.00	1.00	14.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.om.map.-MITListMapBuilder	false	1.00	0.75	25.00	1.00	9.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-DependTypeMapBuilder	false	1.00	0.75	35.00	1.00	19.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-TransitionMapBuilder																		

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.map.-ScopeMapBuilder	false	1.00	0.75	24.00	1.00	8.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	5.00	3.00
org.tigris.-scarab.om.map.-AttributeValueMapBuilder	false	1.00	0.75	39.00	1.00	23.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.om.map.-AttributeMapBuilder	false	1.00	0.75	39.00	1.00	23.00	1.00	1.00	3.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	6.00	3.00
org.tigris.-scarab.pipeline.-SetLocaleValve	false	2.00	1.00	45.00	3.00	32.00	2.50	0.00	2.00	0.00	2.00	1.00	0.00	0.00	0.00	1.00	11.00	6.00
org.tigris.-scarab.pipeline.-DetermineTargetValve	false	2.00	0.00	34.00	5.00	25.00	3.00	0.00	1.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	5.00	5.00
org.tigris.-scarab.pipeline.-AnonymousLoginValve	false	2.00	0.33	32.00	4.00	19.00	1.50	2.00	2.00	1.00	1.00	1.00	0.00	0.00	1.00	0.00	7.00	8.00
org.tigris.-scarab.pipeline.-DetermineCharsetValve	false	2.00	0.00	26.00	2.00	14.00	1.50	0.00	2.00	1.00	1.00	1.00	0.00	0.00	1.00	0.00	4.00	4.00
org.tigris.-scarab.pipeline.-DetermineCharsetValve22	false	2.00	0.00	25.00	2.00	13.00	1.50	0.00	2.00	1.00	1.00	1.00	0.00	0.00	1.00	0.00	4.00	4.00
org.tigris.-scarab.pipeline.-SetInfoMessageValve	false	2.00	0.00	17.00	1.00	4.00	1.00	0.00	1.00	0.00	1.50	1.00	1.00	0.00	0.00	0.00	5.00	2.00
org.tigris.-scarab.pipeline.-FreshenUserValve	false	2.00	0.67	180.00	7.00	160.00	3.00	0.00	4.00	1.00	1.50	1.00	0.00	0.00	0.50	1.00	18.00	28.00
org.tigris.-scarab.pipeline.-TimingInfoValve	false	2.00	0.67	52.00	3.00	37.00	2.50	0.00	2.00	1.00	1.00	3.00	0.00	0.00	1.00	1.00	9.00	6.00
org.tigris.-scarab.pipeline.-ResetCacheValve	false	2.00	0.00	11.00	1.00	2.00	1.00	0.00	1.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	3.00	1.00
org.tigris.-scarab.reports.-ReportDateRange	false	1.00	0.80	33.00	1.17	6.00	1.00	3.00	6.00	0.00	0.50	0.00	0.00	0.00	0.00	1.00	3.00	7.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.reports.-ReportAxis	false	1.00	0.70	55.00	1.83	29.00	1.50	2.00	6.00	0.00	0.67	0.00	0.00	0.00	0.00	4.00	7.00	11.00
org.tigris.-scarab.reports.-ModuleIssueType	false	1.00	0.67	53.00	1.75	20.00	1.25	3.00	8.00	2.00	0.50	0.00	0.00	0.00	0.25	4.00	6.00	14.00
org.tigris.-scarab.reports.-ReportOptionAttribute	false	1.00	0.60	38.00	1.67	12.00	1.17	2.00	6.00	2.00	0.50	0.00	0.00	0.00	0.33	8.00	6.00	10.00
org.tigris.-scarab.reports.-ReportHeading	false	1.00	0.89	380.00	3.76	293.00	2.08	6.00	25.00	0.00	0.48	0.00	0.00	0.00	0.00	6.00	15.00	94.00
org.tigris.-scarab.reports.-ReportUserAttribute	false	1.00	0.62	58.00	1.67	22.00	1.22	3.00	9.00	3.00	0.44	0.00	0.00	0.00	0.33	8.00	6.00	15.00
org.tigris.-scarab.reports.-IncompatibleMITListException	false	6.00	0.00	25.00	1.00	5.00	1.00	0.00	5.00	0.00	1.80	0.00	0.00	0.00	0.00	2.00	5.00	5.00
org.tigris.-scarab.reports.-ReportGroup	false	1.00	0.79	77.00	1.46	28.00	1.23	4.00	13.00	2.00	0.54	0.00	0.00	0.00	0.15	7.00	9.00	19.00
org.tigris.-scarab.reports.-ReportDate	false	1.00	0.62	28.00	1.20	5.00	1.00	2.00	5.00	0.00	0.40	0.00	0.00	0.00	0.00	6.00	4.00	6.00
org.tigris.-scarab.reports.-ReportBridge	false	1.00	0.81	528.00	2.13	362.00	1.64	5.00	45.00	0.00	0.60	0.00	0.00	0.00	0.00	10.00	35.00	96.00
org.tigris.-scarab.reports.-ReportDefinition	false	1.00	0.95	429.00	3.38	338.00	2.28	6.00	29.00	0.00	0.55	1.00	0.00	0.00	0.00	3.00	21.00	98.00
org.tigris.-scarab.reports.-ReportTableModel	false	2.00	0.95	414.00	3.22	294.00	2.04	10.00	23.00	0.00	1.17	10.00	0.00	0.00	0.00	2.00	24.00	74.00
org.tigris.-scarab.screens.-SaveTemplate	false	6.00	0.00	33.00	3.00	23.00	3.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	8.00	3.00
org.tigris.scarab.-screens.Register	false	6.00	0.00	13.00	3.00	8.00	2.00	0.00	1.00	1.00	2.00	0.00	0.00	0.00	6.00	0.00	6.00	3.00
org.tigris.-scarab.screens.-ScarabDefault	false	6.00	0.00	11.00	1.00	3.00	1.00	0.00	1.00	1.00	2.00	0.00	0.00	3.00	6.00	3.00	4.00	1.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.screens.-IssueListExport	false	7.00	0.67	98.00	4.25	74.00	2.50	2.00	4.00	1.00	3.50	0.00	0.00	0.00	1.75	0.00	16.00	17.00
org.tigris.-scarab.screens.-MoveIssue2	false	7.00	0.00	9.00	1.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00	0.00	0.00	7.00	0.00	2.00	1.00
org.tigris.-scarab.screens.-RSSDataExport	false	4.00	1.00	93.00	3.25	68.00	1.75	0.00	4.00	1.00	1.25	9.00	0.00	0.00	1.00	0.00	24.00	13.00
org.tigris.-scarab.screens.-SelectModule	false	6.00	0.00	10.00	1.00	2.00	1.00	0.00	1.00	1.00	2.00	0.00	0.00	0.00	6.00	0.00	4.00	1.00
org.tigris.scarab.-screens.IssueList	false	6.00	0.00	18.00	3.00	10.00	2.00	0.00	1.00	1.00	2.00	0.00	0.00	0.00	6.00	0.00	4.00	3.00
org.tigris.-scarab.screens.-MoveIssue	false	6.00	0.90	60.00	1.75	31.00	1.50	2.00	4.00	2.00	2.00	5.00	0.00	1.00	3.00	1.00	9.00	7.00
org.tigris.-scarab.screens.-ViewAttachment	false	6.00	0.00	51.00	5.00	43.00	4.00	0.00	1.00	1.00	2.00	0.00	0.00	0.00	6.00	0.00	12.00	5.00
org.tigris.scarab.-screens.Confirm	false	7.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
org.tigris.scarab.-screens.Logout	false	7.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
org.tigris.-scarab.screens.-ReportExport	false	7.00	1.00	184.00	16.50	170.00	3.50	0.00	2.00	1.00	3.00	1.00	0.00	0.00	3.50	0.00	22.00	33.00
org.tigris.-scarab.screens.-AssignIssue	false	6.00	0.00	26.00	3.00	16.00	2.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	7.00	3.00
org.tigris.scarab.-screens.Default	false	5.00	0.00	155.00	2.50	119.00	1.60	0.00	5.00	2.00	1.20	0.00	5.00	32.00	2.00	26.00	10.00	25.00
org.tigris.scarab.-screens.Login	false	7.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
org.tigris.-scarab.screens.-DataExport	false	6.00	1.20	120.00	1.83	36.00	1.50	0.00	5.00	1.00	1.17	1.00	1.00	2.00	1.20	2.00	9.00	11.00
org.tigris.-scarab.screens.-DataExport$TSVPrinter		1.00	0.62	59.00	3.00	39.00	2.00	2.00	5.00	0.00	0.80	0.00	0.00	0.00	0.00	2.00	9.00	15.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.screens.-ModuleSwitchingLink	false	4.00	0.00	39.00	1.33	23.00	1.33	1.00	3.00	2.00	1.67	0.00	0.00	0.00	2.67	3.00	10.00	4.00
org.tigris.-scarab.screens.-ViewIssue	false	6.00	0.00	37.00	4.00	27.00	3.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	8.00	4.00
org.tigris.scarab.-screens.admin.-ViewXMLExportIssues	false	6.00	0.00	116.00	19.00	108.00	4.00	0.00	1.00	1.00	2.00	0.00	0.00	0.00	6.00	0.00	14.00	19.00
org.tigris.scarab.-screens.admin.-ArtifactTypeEdit	false	6.00	0.00	12.00	1.00	2.00	1.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	6.00	1.00
org.tigris.scarab.-screens.admin.-GlobalArtifactTypeEdit	false	6.00	0.00	22.00	2.00	12.00	2.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	7.00	2.00
org.tigris.scarab.-screens.admin.-GlobalAttributeEdit	false	6.00	0.00	22.00	2.00	12.00	2.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	7.00	2.00
org.tigris.scarab.-screens.admin.-XMLImportIssuesResults	false	6.00	1.00	100.00	4.67	79.00	3.00	0.00	3.00	2.00	1.33	6.00	0.00	0.00	4.00	0.00	14.00	14.00
org.tigris.scarab.-screens.admin.-AttributeGroupEdit	false	6.00	0.00	22.00	2.00	12.00	2.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	7.00	2.00
org.tigris.scarab.-screens.admin.-ActivityList	false	6.00	0.00	113.00	6.50	101.00	3.50	0.00	2.00	1.00	1.50	0.00	0.00	0.00	3.00	0.00	13.00	13.00
org.tigris.scarab.-screens.admin.-ViewXMLExportSettings	false	6.00	0.00	46.00	10.00	38.00	3.00	0.00	1.00	1.00	2.00	0.00	0.00	0.00	6.00	0.00	5.00	10.00
org.tigris.scarab.-screens.admin.-UserAttributeEdit	false	6.00	0.00	22.00	2.00	12.00	2.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	7.00	2.00
org.tigris.scarab.-screens.admin.-ConditionEdit	false	6.00	0.00	12.00	1.00	2.00	1.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	6.00	1.00
org.tigris.scarab.-screens.admin.-ManageArtifactTypes	false	6.00	0.00	12.00	1.00	2.00	1.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	6.00	1.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-screens.admin.-AttributeOptionSelect	false	6.00	0.00	12.00	1.00	2.00	1.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	6.00	1.00
org.tigris.scarab.-screens.admin.-ModuleAttributeEdit	false	6.00	0.00	45.00	5.00	35.00	4.00	0.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	10.00	5.00
org.tigris.scarab.-screens.entry.-Wizard3	false	7.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
org.tigris.scarab.-screens.entry.-Wizard1	false	6.00	0.00	26.00	2.00	16.00	2.00	0.00	1.00	0.00	4.00	0.00	0.00	1.00	0.00	1.00	9.00	2.00
org.tigris.scarab.-screens.home.-XModuleList	false	6.00	0.00	15.00	3.00	7.00	2.00	0.00	1.00	1.00	2.00	0.00	0.00	0.00	6.00	0.00	5.00	3.00
org.tigris.-scarab.services.-TorqueService	false	2.00	0.00	72.00	1.00	56.00	1.00	0.00	3.00	1.00	0.00	0.00	0.00	0.00	0.67	0.00	3.00	3.00
org.tigris.-scarab.services.-DatabaseInitializer	false	2.00	1.00	256.00	4.40	214.00	3.20	0.00	5.00	0.00	0.60	9.00	0.00	0.00	0.00	0.00	16.00	22.00
org.tigris.scarab.-services.cache.-DefaultScarabCacheService	false	2.00	0.86	222.00	1.87	159.00	1.87	4.00	15.00	0.00	2.40	0.00	0.00	0.00	0.00	0.00	19.00	28.00
org.tigris.scarab.-services.cache.-ScarabCache	false	1.00	0.92	59.00	1.14	24.00	1.14	0.00	0.00	0.00	2.57	1.00	14.00	0.00	0.00	0.00	8.00	16.00
org.tigris.scarab.-services.cache.-ScarabCacheKey	false	1.00	0.79	180.00	2.00	109.00	1.94	8.00	16.00	2.00	1.94	1.00	0.00	0.00	0.12	2.00	6.00	32.00
org.tigris.scarab.-services.cache.-NoOpScarabCacheService	false	1.00	0.00	51.00	1.00	7.00	1.00	0.00	15.00	0.00	3.47	0.00	0.00	0.00	0.00	0.00	5.00	15.00
org.tigris.scarab.-services.cache.-ScarabCacheService	false	0.00	0.00	25.00	1.00	21.00	1.00	0.00	12.00	0.00	2.92	0.00	0.00	2.00	0.00	3.00	4.00	12.00
org.tigris.scarab.-services.email.-VelocityEmailService	false	3.00	0.95	304.00	2.21	227.00	1.71	1.00	14.00	0.00	2.64	4.00	0.00	0.00	0.00	3.00	25.00	31.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-services.email.-VelocityEmail	false	1.00	0.00	46.00	1.00	9.00	1.00	0.00	1.00	0.00	3.00	0.00	6.00	0.00	0.00	0.00	5.00	7.00
org.tigris.scarab.-services.email.-EmailService	false	0.00	1.20	23.00	1.00	18.00	1.00	1.00	6.00	0.00	3.50	0.00	0.00	1.00	0.00	1.00	5.00	6.00
org.tigris.-scarab.services.-security.-ScarabSecurity	false	2.00	1.00	132.00	1.45	48.00	1.27	2.00	7.00	0.00	0.64	24.00	4.00	1.00	0.00	2.00	7.00	16.00
org.tigris.-scarab.services.-security.-ScarabDBSecurityService	false	4.00	0.00	97.00	2.33	69.00	2.17	0.00	6.00	6.00	1.17	0.00	0.00	0.00	4.00	0.00	13.00	14.00
org.tigris.-scarab.tools.-ScarabRequestTool	true	1.00	0.97	2227.00	2.12	1629.00	1.58	34.00	145.00	0.00	0.81	1.00	0.00	0.00	0.00	58.00	74.00	307.00
org.tigris.-scarab.tools.-ScarabRequestTool$IssueListIterator		1.00	0.68	54.00	1.80	28.00	1.40	4.00	5.00	0.00	0.80	3.00	0.00	0.00	0.00	2.00	74.00	9.00
org.tigris.-scarab.tools.-SecurityAdminTool	true	1.00	1.00	161.00	2.07	110.00	1.64	0.00	14.00	0.00	0.71	2.00	0.00	0.00	0.00	4.00	19.00	29.00
org.tigris.-scarab.tools.-ScarabGlobalTool	true	1.00	0.99	413.00	1.52	218.00	1.26	3.00	52.00	0.00	1.09	3.00	2.00	0.00	0.00	4.00	23.00	82.00
org.tigris.-scarab.tools.-ScarabLocalizationTool	true	2.00	0.94	333.00	2.17	285.00	1.79	5.00	29.00	5.00	1.10	4.00	0.00	1.00	0.34	76.00	19.00	63.00
org.tigris.-scarab.tools.-ScarabToolManager	true	1.00	0.50	57.00	4.67	46.00	2.67	1.00	3.00	0.00	2.00	0.00	0.00	0.00	0.00	7.00	11.00	14.00
org.tigris.scarab.-tools.Format	true	1.00	1.33	22.00	1.00	5.00	1.00	0.00	0.00	0.00	1.50	1.00	4.00	0.00	0.00	0.00	4.00	4.00
org.tigris.-scarab.tools.-localization.-L10NMessage	true	1.00	0.83	58.00	1.57	36.00	1.29	2.00	7.00	0.00	1.86	0.00	0.00	0.00	0.00	25.00	6.00	11.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.tools.-localization.-L10NKeySet	true	1.00	0.00	350.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	346.00	0.00	0.00	0.00	0.00	2.00	0.00
org.tigris.-scarab.tools.-localization.-Localizable	true	0.00	0.00	6.00	1.00	2.00	1.00	0.00	2.00	0.00	0.50	0.00	0.00	13.00	0.00	20.00	2.00	2.00
org.tigris.-scarab.tools.-localization.-LocalizationKey	true	0.00	0.00	5.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00	17.00	2.00	1.00
org.tigris.-scarab.tools.-localization.-L10NKey	true	1.00	0.67	23.00	1.00	6.00	1.00	1.00	4.00	1.00	0.50	0.00	0.00	0.00	0.25	1.00	4.00	4.00
org.tigris.scarab.-util.OptionModel	true	1.00	0.80	52.00	1.00	12.00	1.00	3.00	11.00	0.00	1.18	0.00	0.00	0.00	0.00	0.00	1.00	11.00
org.tigris.-scarab.util.-SimpleHandlerClient	true	1.00	0.00	29.00	1.50	19.00	1.50	0.00	1.00	0.00	0.50	0.00	1.00	0.00	0.00	1.00	8.00	3.00
org.tigris.-scarab.util.-ScarabRuntimeException	true	4.00	0.89	116.00	1.40	78.00	1.33	2.00	15.00	1.00	2.20	0.00	0.00	0.00	0.27	8.00	10.00	21.00
org.tigris.-scarab.util.-AnonymousUserUtil	true	1.00	0.00	73.00	2.80	54.00	2.20	0.00	0.00	0.00	0.40	0.00	5.00	0.00	0.00	0.00	6.00	14.00
org.tigris.-scarab.util.-SubsetIteratorWithSize	false	2.00	0.50	20.00	1.00	5.00	1.00	1.00	3.00	0.00	1.67	0.00	0.00	0.00	0.00	4.00	2.00	3.00
org.tigris.-scarab.util.-RegexProcessor	false	1.00	0.67	30.00	2.00	17.00	1.50	1.00	2.00	0.00	1.50	2.00	0.00	0.00	0.00	1.00	10.00	4.00
org.tigris.-scarab.util.-ComponentLocator	false	1.00	0.50	25.00	1.67	15.00	1.67	0.00	0.00	0.00	0.67	1.00	3.00	0.00	0.00	0.00	6.00	5.00
org.tigris.-scarab.util.-ScarabLocalizedTorqueException	false	5.00	0.62	34.00	1.20	23.00	1.20	2.00	5.00	2.00	0.60	0.00	0.00	0.00	2.00	2.00	5.00	6.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-util.ScarabLink	true	3.00	0.92	413.00	2.74	302.00	1.77	11.00	31.00	9.00	0.58	0.00	0.00	2.00	0.87	10.00	17.00	85.00
org.tigris.scarab.-util.TableModel	true	1.00	0.89	95.00	1.00	11.00	1.00	2.00	7.00	0.00	0.70	0.00	3.00	1.00	0.00	1.00	5.00	10.00
org.tigris.-scarab.util.-TableModel$ColumnHeading		2.00	0.00	14.00	3.00	7.00	2.00	0.00	1.00	1.00	0.00	0.00	0.00	0.00	2.00	4.00	5.00	3.00
org.tigris.-scarab.util.-TableModel$RowHeading		2.00	0.00	14.00	3.00	7.00	2.00	0.00	1.00	1.00	0.00	0.00	0.00	0.00	2.00	0.00	5.00	3.00
org.tigris.-scarab.util.-TableModel$Heading		1.00	0.80	29.00	1.00	6.00	1.00	2.00	6.00	0.00	0.33	0.00	0.00	2.00	0.00	4.00	5.00	6.00
org.tigris.scarab.-util.Log	false	1.00	1.00	24.00	1.00	9.00	1.00	0.00	0.00	0.00	0.33	1.00	3.00	0.00	0.00	0.00	3.00	3.00
org.tigris.-scarab.util.-ScarabLocalizedTurbineSecurityException	false	6.00	0.62	35.00	1.20	24.00	1.20	2.00	5.00	2.00	0.60	0.00	0.00	0.00	2.40	1.00	5.00	6.00
org.tigris.-scarab.util.-EmailContext	true	2.00	0.88	42.00	1.00	9.00	1.11	1.00	9.00	0.00	0.78	0.00	0.00	0.00	0.00	8.00	8.00	9.00
org.tigris.-scarab.util.-ValidationException	true	6.00	0.00	36.00	1.00	8.00	1.00	0.00	8.00	0.00	2.25	0.00	0.00	0.00	0.00	1.00	5.00	8.00
org.tigris.-scarab.util.-ScarabPaginatedList	true	1.00	0.80	77.00	1.25	33.00	1.25	4.00	12.00	0.00	0.58	0.00	0.00	0.00	0.00	4.00	1.00	15.00
org.tigris.-scarab.util.-TurbineInitialization	false	1.00	0.50	39.00	2.00	23.00	1.33	0.00	0.00	0.00	1.33	1.00	3.00	0.00	0.00	0.00	6.00	6.00
org.tigris.-scarab.util.-ReferenceInsertionFilter	true	1.00	0.00	71.00	5.67	57.00	2.33	0.00	2.00	0.00	1.67	0.00	1.00	0.00	0.00	0.00	6.00	17.00
org.tigris.-scarab.util.-SimpleSkipFiltering	true	1.00	0.00	14.00	1.00	2.00	1.00	1.00	2.00	1.00	0.50	0.00	0.00	0.00	0.50	4.00	3.00	2.00
org.tigris.scarab.-util.Email	true	2.00	0.92	309.00	3.50	253.00	2.07	0.00	1.00	1.00	3.29	2.00	13.00	0.00	2.00	3.00	26.00	49.00
org.tigris.scarab.-util.AntServlet	false	3.00	0.00	97.00	5.00	80.00	3.00	0.00	2.00	2.00	1.50	2.00	0.00	0.00	3.00	0.00	12.00	10.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-util.IssueIdParser	false	1.00	1.00	112.00	3.20	76.00	2.40	0.00	0.00	0.00	1.80	1.00	5.00	0.00	0.00	0.00	12.00	16.00
org.tigris.-scarab.util.-ScarabConstants	true	0.00	0.00	76.00	0.00	0.00	0.00	58.00	0.00	0.00	0.00	8.00	0.00	0.00	0.00	0.00	3.00	0.00
org.tigris.scarab.-util.ScarabUtil	true	1.00	0.96	158.00	4.80	101.00	3.40	0.00	0.00	0.00	1.80	6.00	5.00	0.00	0.00	0.00	9.00	24.00
org.tigris.scarab.-util.EmailLink	true	1.00	0.96	281.00	1.70	162.00	1.52	10.00	33.00	1.00	0.94	2.00	0.00	1.00	0.03	5.00	12.00	56.00
org.tigris.scarab.-util.SkipFiltering	true	0.00	0.00	5.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	7.00	0.00	7.00	1.00	1.00
org.tigris.-scarab.util.-MutableBoolean	true	1.00	0.50	17.00	1.00	3.00	1.00	1.00	3.00	0.00	0.67	0.00	0.00	0.00	0.00	3.00	0.00	3.00
org.tigris.-scarab.util.-SubsetIterator	false	1.00	0.80	61.00	1.83	31.00	1.67	6.00	6.00	0.00	0.83	1.00	0.00	2.00	0.00	4.00	7.00	11.00
org.tigris.-scarab.util.-EmptySubsetIterator		2.00	0.00	20.00	1.00	5.00	1.00	0.00	4.00	3.00	0.00	0.00	0.00	0.00	1.50	2.00	7.00	4.00
org.tigris.-scarab.util.-ScarabException	true	5.00	0.86	85.00	1.13	55.00	1.13	1.00	15.00	1.00	2.20	0.00	0.00	5.00	0.33	45.00	8.00	17.00
org.tigris.-scarab.util.-WindowIterator	true	1.00	0.72	96.00	2.71	99.00	1.86	7.00	7.00	0.00	0.71	2.00	0.00	1.00	0.00	4.00	9.00	19.00
org.tigris.-scarab.util.-EmptyWindowIterator		2.00	0.00	28.00	1.00	7.00	1.00	0.00	6.00	5.00	0.33	0.00	0.00	0.00	1.67	2.00	9.00	6.00
org.tigris.scarab.-util.StaticLink	true	1.00	0.74	52.00	1.00	14.00	1.00	3.00	10.00	1.00	0.20	0.00	0.00	0.00	0.10	1.00	7.00	10.00
org.tigris.-scarab.util.-SimpleHandler	true	1.00	1.00	235.00	4.00	189.00	2.38	0.00	6.00	0.00	3.25	2.00	2.00	0.00	0.00	1.00	27.00	32.00
org.tigris.-scarab.util.-PasswordGenerator	true	1.00	0.00	11.00	1.00	3.00	1.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00	0.00	1.00	1.00
org.tigris.-scarab.util.-EmailLinkFactory	true	1.00	0.00	84.00	3.00	20.00	3.00	0.00	0.00	0.00	1.00	2.00	1.00	0.00	0.00	0.00	7.00	3.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.util.-EmailLinkFactory$ErrorEmailLink		2.00	0.00	27.00	1.67	14.00	1.33	0.00	3.00	1.00	0.33	0.00	0.00	0.00	0.67	2.00	7.00	5.00
org.tigris.-scarab.util.-IteratorWithSize	true	0.00	0.00	6.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	1.00	0.00	4.00	0.00	15.00	6.00	1.00
org.tigris.-scarab.util.-EmptyIteratorWithSize		1.00	0.00	20.00	1.00	5.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	6.00	4.00
org.tigris.-scarab.util.-SnippetRenderer	true	1.00	0.50	31.00	1.50	16.00	1.50	2.00	2.00	1.00	1.00	2.00	0.00	0.00	0.50	3.00	6.00	3.00
org.tigris.-scarab.util.build.-AntPropertyFileGenerator	false	3.00	0.25	66.00	2.60	55.00	2.00	1.00	5.00	1.00	1.00	0.00	0.00	0.00	0.60	0.00	8.00	13.00
org.tigris.-scarab.util.build.-AntSchemaFix	false	3.00	0.83	112.00	2.71	84.00	1.86	3.00	7.00	1.00	0.86	0.00	0.00	0.00	0.43	0.00	13.00	19.00
org.tigris.-scarab.util.build.-AntL10AnalysisTask	false	3.00	0.86	234.00	1.44	172.00	1.22	7.00	9.00	1.00	0.78	0.00	0.00	0.00	0.33	0.00	17.00	13.00
org.tigris.-scarab.util.build.-AntL10AnalysisTask$Message		1.00	1.00	29.00	2.00	17.00	1.33	2.00	3.00	0.00	0.67	0.00	0.00	0.00	0.00	2.00	17.00	6.00
org.tigris.-scarab.util.build.-PropertyGetter	false	0.00	0.00	5.00	1.00	1.00	1.00	0.00	1.00	0.00	2.00	0.00	0.00	2.00	0.00	3.00	2.00	1.00
org.tigris.-scarab.util.build.-PropertyFileGenerator	true	1.00	0.79	200.00	4.22	186.00	2.78	3.00	9.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	16.00	38.00
org.tigris.-scarab.util.build.-l10nchecker.-L10nIssue	true	1.00	1.04	43.00	1.12	23.00	1.12	0.00	8.00	0.00	0.12	4.00	0.00	12.00	0.00	14.00	3.00	9.00
org.tigris.-scarab.util.build.-l10nchecker.-L10nKey	true	1.00	0.88	66.00	1.08	20.00	1.08	6.00	12.00	2.00	0.58	0.00	0.00	0.00	0.17	3.00	4.00	13.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.util.build.-l10nchecker.-L10nInspector	true	1.00	0.87	406.00	5.82	356.00	3.00	5.00	10.00	0.00	0.64	8.00	1.00	0.00	0.00	3.00	32.00	64.00
org.tigris.-scarab.util.build.-l10nchecker.-L10nIssueTemplates	true	1.00	0.00	25.00	2.00	17.00	1.67	0.00	0.00	0.00	1.00	1.00	3.00	0.00	0.00	0.00	4.00	6.00
org.tigris.-scarab.util.build.-l10nchecker.-L10nMessage	true	1.00	0.82	42.00	1.00	10.00	1.00	4.00	8.00	0.00	0.50	0.00	0.00	0.00	0.00	3.00	4.00	8.00
org.tigris.-scarab.util.build.-l10nchecker.-issues.-NotTranslatedIssue	true	2.00	0.50	17.00	1.00	3.00	1.00	1.00	3.00	0.00	0.33	0.00	0.00	0.00	0.00	1.00	3.00	3.00
org.tigris.-scarab.util.build.-l10nchecker.-issues.-NotInReferenceIssue	true	2.00	0.50	17.00	1.00	3.00	1.00	1.00	3.00	0.00	0.33	0.00	0.00	0.00	0.00	1.00	3.00	3.00
org.tigris.-scarab.util.build.-l10nchecker.-issues.-DefinedTwiceDiffIssue	true	2.00	0.50	19.00	1.00	4.00	1.00	2.00	3.00	0.00	0.67	0.00	0.00	0.00	0.00	0.00	4.00	3.00
org.tigris.-scarab.util.build.-l10nchecker.-issues.-DefinedTwiceIssue	true	2.00	0.50	19.00	1.00	4.00	1.00	2.00	3.00	0.00	0.67	0.00	0.00	0.00	0.00	1.00	4.00	3.00
org.tigris.-scarab.util.build.-l10nchecker.-issues.-TranslationMissingIssue	true	2.00	0.50	17.00	1.00	3.00	1.00	1.00	3.00	0.00	0.33	0.00	0.00	0.00	0.00	1.00	3.00	3.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.util.build.-l10nchecker.-issues.-CantParseLineIssue	true	2.00	0.50	17.00	1.00	3.00	1.00	1.00	3.00	0.00	0.33	0.00	0.00	0.00	0.00	1.00	3.00	3.00
org.tigris.-scarab.util.build.-l10nchecker.-issues.-IllegalPatternIssue	true	2.00	0.50	21.00	1.00	5.00	1.00	3.00	3.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	4.00	3.00
org.tigris.-scarab.util.build.-l10nchecker.-issues.-DifferentAttributeCountIssue	true	2.00	0.50	17.00	1.00	3.00	1.00	1.00	3.00	0.00	0.33	0.00	0.00	0.00	0.00	1.00	3.00	3.00
org.tigris.-scarab.util.build.-l10nchecker.-issues.-NoTransAllowedIssue	true	2.00	0.50	17.00	1.00	3.00	1.00	1.00	3.00	0.00	0.33	0.00	0.00	0.00	0.00	1.00	3.00	3.00
org.tigris.-scarab.util.build.-l10nchecker.-issues.-TranslationRequiredIssue	true	2.00	0.50	19.00	1.00	4.00	1.00	2.00	3.00	0.00	0.67	0.00	0.00	0.00	0.00	1.00	4.00	3.00
org.tigris.-scarab.util.build.-l10nchecker.-issues.-TranslatedTwiceIssue	true	2.00	0.50	19.00	1.00	4.00	1.00	2.00	3.00	0.00	0.67	0.00	0.00	0.00	0.00	1.00	4.00	3.00
org.tigris.-scarab.util.build.-l10nchecker.-issues.-TranslatedTwiceDiffIssue	true	1.00	0.00	13.00	1.00	2.00	1.00	1.00	2.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	3.00	2.00
org.tigris.scarab.-util.comparators.-StartsWithPredicate	true	1.00	0.00	13.00	1.00	2.00	1.00	1.00	2.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	3.00	2.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-util.export.-ExportFormat	true	1.00	0.00	7.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.00	0.00	0.00	0.00	0.00	1.00	0.00
org.tigris.scarab.-util.word.-SearchIndex	true	0.00	1.14	29.00	1.00	15.00	1.00	9.00	8.00	0.00	0.88	0.00	0.00	1.00	0.00	11.00	5.00	8.00
org.tigris.-scarab.util.word.-PorterStemAnalyzer	true	2.00	0.00	36.00	2.00	33.00	5.00	0.00	1.00	1.00	2.00	1.00	0.00	0.00	2.00	1.00	7.00	2.00
org.tigris.-scarab.util.word.-IssueSearchFactory	true	1.00	0.86	112.00	2.25	75.00	2.12	3.00	8.00	0.00	0.88	1.00	0.00	2.00	0.00	3.00	9.00	18.00
org.tigris.-scarab.util.word.-MaxConcurrentSearchException	true	6.00	0.00	36.00	1.00	8.00	1.00	0.00	8.00	0.00	2.25	0.00	0.00	0.00	0.00	7.00	5.00	8.00
org.tigris.scarab.-util.word.-SearchFactory	true	1.00	0.00	35.00	1.00	22.00	1.00	0.00	0.00	0.00	0.67	1.00	3.00	0.00	0.00	0.00	5.00	3.00
org.tigris.scarab.-util.word.-QueryResult	true	1.00	0.91	123.00	1.41	58.00	1.35	8.00	17.00	0.00	0.41	0.00	0.00	0.00	0.00	10.00	9.00	24.00
org.tigris.scarab.-util.word.-IssueSearch	true	4.00	0.99	2078.00	3.58	1368.00	2.04	38.00	90.00	1.00	0.98	46.00	0.00	0.00	0.04	18.00	61.00	322.00
org.tigris.-scarab.util.word.-IssueSearch$QueryResultIterator		1.00	0.83	253.00	2.70	213.00	2.40	7.00	10.00	0.00	1.30	0.00	0.00	0.00	0.00	4.00	61.00	27.00
org.tigris.-scarab.util.word.-IssueSearch$ColumnBundle		1.00	0.00	7.00	0.00	0.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	61.00	0.00
org.tigris.-scarab.util.word.-IssueSearch$ResultSetAndSize		1.00	0.00	10.00	1.00	2.00	1.00	2.00	1.00	0.00	2.00	0.00	0.00	0.00	0.00	4.00	61.00	1.00
org.tigris.-scarab.util.word.-LuceneSearchIndex	false	1.00	0.87	505.00	3.82	432.00	2.47	6.00	17.00	0.00	1.18	1.00	0.00	0.00	0.00	0.00	36.00	65.00
org.tigris.-scarab.util.word.-ComplexQueryException	true	6.00	0.00	24.00	1.00	5.00	1.00	0.00	5.00	0.00	1.80	0.00	0.00	0.00	0.00	8.00	5.00	5.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-util.xmlissues.-XmlModule	true	1.00	0.89	79.00	1.00	16.00	1.00	8.00	17.00	0.00	0.47	0.00	0.00	0.00	0.00	3.00	3.00	17.00
org.tigris.scarab.-util.xmlissues.-XmlAttachment	true	1.00	0.96	115.00	1.00	24.00	1.00	12.00	25.00	0.00	0.48	0.00	0.00	0.00	0.00	3.00	4.00	25.00
org.tigris.scarab.-util.xmlissues.-ImportErrors	true	2.00	0.62	34.00	1.80	23.00	1.20	2.00	5.00	0.00	0.40	0.00	0.00	0.00	0.00	3.00	5.00	9.00
org.tigris.scarab.-util.xmlissues.-CreatedDate	true	2.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	1.00	0.00
org.tigris.scarab.-util.xmlissues.-ModifiedDate	true	2.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	1.00	0.00
org.tigris.scarab.-util.xmlissues.-ImportIssues	true	1.00	0.96	334.00	1.59	221.00	1.38	9.00	32.00	0.00	1.00	4.00	0.00	0.00	0.00	4.00	32.00	51.00
org.tigris.scarab.-util.xmlissues.-ImportIssues$ScarabIssuesSetupRule		2.00	0.00	10.00	1.00	4.00	1.00	0.00	1.00	1.00	3.00	0.00	0.00	0.00	2.00	2.00	32.00	1.00
org.tigris.scarab.-util.xmlissues.-XmlIssue	true	1.00	0.88	44.00	1.29	15.00	1.29	3.00	7.00	0.00	0.43	1.00	0.00	0.00	0.00	2.00	8.00	9.00
org.tigris.scarab.-util.xmlissues.-BaseDate	true	1.00	0.67	53.00	1.00	36.00	1.00	2.00	6.00	1.00	0.33	1.00	0.00	3.00	0.17	4.00	5.00	6.00
org.tigris.scarab.-util.xmlissues.-Dependency	true	1.00	0.78	67.00	1.07	16.00	1.00	5.00	14.00	2.00	0.43	0.00	0.00	0.00	0.14	3.00	3.00	15.00
org.tigris.scarab.-util.xmlissues.-EndDate	true	2.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00
org.tigris.scarab.-util.xmlissues.-ImportIssuesTask	true	4.00	0.00	61.00	1.08	19.00	1.08	1.00	12.00	1.00	0.42	0.00	0.00	0.00	0.33	0.00	6.00	13.00
org.tigris.scarab.-util.xmlissues.-ScarabIssues	true	1.00	0.90	1011.00	6.42	931.00	2.67	13.00	24.00	0.00	1.12	5.00	0.00	0.00	0.00	4.00	40.00	154.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-util.xmlissues.-XmlActivitySet	true	1.00	0.91	83.00	1.43	31.00	1.29	6.00	14.00	0.00	0.43	0.00	0.00	0.00	0.00	2.00	7.00	20.00
org.tigris.scarab.-util.xmlissues.-XmlActivity	true	1.00	0.96	169.00	1.00	45.00	1.00	15.00	35.00	1.00	0.40	0.00	0.00	0.00	0.03	2.00	5.00	35.00
org.tigris.scarab.-workflow.-Workflow	true	0.00	0.00	63.00	1.00	59.00	1.00	0.00	15.00	0.00	3.73	0.00	0.00	2.00	0.00	3.00	11.00	15.00
org.tigris.scarab.-workflow.-CheapWorkflow	true	2.00	0.00	114.00	10.50	104.00	7.00	0.00	2.00	1.00	3.00	0.00	0.00	0.00	1.00	0.00	18.00	21.00
org.tigris.scarab.-workflow.-DefaultWorkflow	true	1.00	0.00	101.00	1.00	7.00	1.00	0.00	15.00	0.00	3.73	0.00	0.00	1.00	0.00	2.00	12.00	15.00
org.tigris.scarab.-workflow.-WorkflowFactory	true	1.00	0.50	50.00	1.00	47.00	1.00	0.00	0.00	0.00	0.33	1.00	3.00	0.00	0.00	0.00	10.00	3.00
org.tigris.scarab.-migration.-JDBCTask	false	3.00	0.92	181.00	1.46	119.00	1.17	10.00	23.00	0.00	0.50	1.00	1.00	1.00	0.00	1.00	16.00	35.00
org.tigris.scarab.-migration.-b15b16.-DB_1_MoveIssueCreateInfo	false	4.00	1.08	270.00	2.62	213.00	2.25	3.00	8.00	1.00	1.12	6.00	0.00	0.00	0.50	0.00	13.00	21.00
org.tigris.scarab.-migration.-b18b19.-MigrateProperties	false	3.00	0.50	146.00	1.00	41.00	1.00	1.00	2.00	1.00	0.50	1.00	0.00	0.00	1.50	0.00	7.00	2.00
org.tigris.scarab.-SecurityTest	false	5.00	0.00	22.00	1.00	15.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	6.00	1.00
org.tigris.scarab.-StartingTorqueTest	false	4.00	0.00	11.00	1.00	4.00	1.00	0.00	2.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	4.00	2.00
org.tigris.scarab.-StartingTurbineTest	false	4.00	0.00	16.00	2.00	10.00	2.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	5.00	2.00
org.tigris.-scarab.actions.-RegisterTest	false	3.00	0.00	13.00	1.00	2.00	1.00	1.00	2.00	1.00	0.00	0.00	0.00	0.00	1.50	1.00	2.00	2.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.da.-AttributeAccessTest	false	5.00	0.00	106.00	2.00	72.00	1.00	0.00	10.00	1.00	0.20	0.00	0.00	0.00	0.50	1.00	6.00	20.00
org.tigris.-scarab.feeds.-IssueFeedTest	false	5.00	0.00	15.00	1.00	10.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	1.00
org.tigris.-scarab.feeds.-QueryFeedTest	false	5.00	0.00	14.00	1.00	9.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	1.00
org.tigris.-scarab.om.-AttributeOptionTest	false	5.00	0.00	123.00	1.44	144.00	1.44	1.00	9.00	1.00	0.00	0.00	0.00	0.00	0.56	1.00	5.00	13.00
org.tigris.-scarab.om.-ScarabModuleTest	false	5.00	0.43	111.00	3.00	81.00	1.38	1.00	8.00	1.00	0.50	0.00	0.00	0.00	0.62	0.00	16.00	24.00
org.tigris.-scarab.om.-RModuleOptionTest	false	5.00	0.75	51.00	1.00	25.00	1.00	4.00	6.00	1.00	0.00	0.00	0.00	0.00	0.83	1.00	7.00	6.00
org.tigris.-scarab.om.-ActivitySetTest	false	5.00	0.00	30.00	1.00	21.00	1.00	1.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	7.00	2.00
org.tigris.-scarab.om.-RModuleIssueTypeTest	false	5.00	0.00	27.00	1.00	10.00	1.00	1.00	4.00	1.00	0.00	0.00	0.00	0.00	1.25	1.00	2.00	4.00
org.tigris.-scarab.om.-AttributeValueTest	false	5.00	0.72	85.00	1.00	43.00	1.00	4.00	11.00	1.00	0.00	0.00	0.00	0.00	0.45	0.00	8.00	11.00
org.tigris.scarab.-om.IssueTest	false	5.00	0.98	205.00	1.62	146.00	1.00	2.00	16.00	4.00	0.12	2.00	0.00	0.00	1.25	1.00	16.00	26.00
org.tigris.scarab.-om.AttributeTest	false	5.00	0.00	29.00	1.50	17.00	1.50	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	4.00	3.00
org.tigris.-scarab.om.-AttachmentTest	false	5.00	0.56	84.00	1.00	57.00	1.00	3.00	7.00	1.00	0.00	0.00	0.00	0.00	0.71	0.00	11.00	7.00
org.tigris.-scarab.om.-RModuleAttributeTest	false	5.00	0.50	35.00	1.00	14.00	1.00	2.00	5.00	1.00	0.00	0.00	0.00	0.00	1.00	1.00	4.00	5.00
org.tigris.scarab.-om.ActivityTest	false	5.00	0.00	49.00	1.00	36.00	1.00	0.00	5.00	2.00	0.00	0.00	0.00	0.00	2.00	1.00	7.00	5.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.-scarab.om.-AttachmentTest2	false	5.00	0.58	57.00	1.00	36.00	1.00	3.00	5.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00	9.00	5.00
org.tigris.-scarab.om.-ScarabUserTest	false	5.00	0.00	25.00	1.00	13.00	1.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	4.00	2.00
org.tigris.-scarab.om.-IssueTypeTest	false	5.00	0.54	102.00	1.00	52.00	1.00	2.00	15.00	1.00	0.07	0.00	0.00	0.00	0.33	1.00	3.00	15.00
org.tigris.-scarab.om.-AttributeGroupTest	false	5.00	0.70	55.00	1.17	41.00	1.17	4.00	6.00	2.00	0.00	0.00	0.00	0.00	1.67	1.00	8.00	7.00
org.tigris.scarab.-om.QueryTest	false	5.00	0.56	131.00	1.33	115.00	1.33	2.00	9.00	1.00	0.00	0.00	0.00	0.00	0.56	1.00	10.00	12.00
org.tigris.-scarab.pipeline.-PipelineCreationTest	false	3.00	1.00	21.00	1.00	11.00	1.00	1.00	2.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	10.00	2.00
org.tigris.scarab.-services.email.-VelocityEmailServiceTest	false	4.00	0.64	64.00	1.33	42.00	1.33	7.00	3.00	1.00	0.33	0.00	0.00	0.00	1.33	1.00	10.00	4.00
org.tigris.scarab.-services.email.-VelocityEmailServiceTest$MockLink		1.00	0.00	5.00	1.00	1.00	1.00	0.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	2.00	10.00	1.00
org.tigris.scarab.-services.hsql.-HSQLServiceTest	false	4.00	0.00	26.00	1.00	14.00	1.00	1.00	4.00	2.00	0.00	0.00	0.00	0.00	2.00	0.00	5.00	4.00
org.tigris.-scarab.services.-yaaficomponent.-YaafiComponentServiceTest	false	4.00	0.00	12.00	1.00	3.00	1.00	1.00	2.00	1.00	0.00	0.00	0.00	0.00	2.00	1.00	2.00	2.00
org.tigris.-scarab.test.-BaseTurbineTestCase	false	3.00	0.90	46.00	1.33	29.00	1.33	0.00	6.00	2.00	0.17	2.00	0.00	31.00	1.00	6.00	7.00	8.00
org.tigris.-scarab.test.-AllScarabTests	false	5.00	0.00	35.00	1.00	30.00	1.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	25.00	1.00
org.tigris.-scarab.test.-BaseScarabTestCase	false	4.00	1.00	120.00	1.53	59.00	1.53	7.00	15.00	2.00	0.07	5.00	0.00	26.00	0.53	27.00	10.00	23.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-test.mocks.-MockScarabLocalizationTool	false	3.00	0.00	6.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	0.00	0.00	0.00	3.00	2.00	2.00	1.00
org.tigris.scarab.-test.mocks.-MockScarabSecurity	false	3.00	0.00	10.00	1.00	2.00	1.00	0.00	2.00	1.00	0.50	0.00	0.00	0.00	1.50	1.00	2.00	2.00
org.tigris.scarab.-test.mocks.-MockScarabLink	false	4.00	0.00	9.00	1.00	2.00	1.00	0.00	2.00	2.00	0.50	0.00	0.00	0.00	4.00	1.00	3.00	2.00
org.tigris.scarab.-test.mocks.-MockServiceBroker	false	1.00	0.00	35.00	1.00	8.00	1.00	0.00	12.00	0.00	0.75	0.00	0.00	0.00	0.00	0.00	6.00	12.00
org.tigris.scarab.-test.mocks.-MockFulcrumServiceManager	false	1.00	0.00	50.00	1.12	15.00	1.06	0.00	16.00	0.00	0.69	0.00	0.00	0.00	0.00	0.00	9.00	18.00
org.tigris.scarab.-test.mocks.-MockSecurityService	false	1.00	0.99	191.00	1.00	46.00	1.00	1.00	69.00	0.00	0.91	0.00	0.00	0.00	0.00	1.00	17.00	69.00
org.tigris.-scarab.tools.-ScarabToolManagerTest	false	5.00	0.00	16.00	1.00	9.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.00	1.00
org.tigris.-scarab.util.-ScarabUtilTest	false	3.00	0.00	13.00	1.00	6.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
org.tigris.-scarab.util.-SubsetIteratorWithSizeTest	false	4.00	0.73	188.00	1.50	127.00	1.12	0.00	8.00	5.00	0.12	7.00	0.00	0.00	2.50	1.00	12.00	12.00
org.tigris.-scarab.util.-SubsetIteratorWithSizeTest$MockIteratorWithSize		1.00	0.50	26.00	1.00	6.00	1.00	2.00	5.00	0.00	0.20	0.00	0.00	0.00	0.00	2.00	12.00	5.00
org.tigris.-scarab.util.-SubsetIteratorTest	false	3.00	0.70	122.00	1.80	81.00	1.30	2.00	10.00	2.00	0.10	5.00	0.00	1.00	0.60	3.00	8.00	18.00
org.tigris.-scarab.util.-EmailLinkTest	false	5.00	0.00	13.00	1.00	6.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	4.00	1.00
org.tigris.-scarab.util.word.-LuceneSearchIndexTest	false	4.00	1.00	12.00	1.00	4.00	1.00	1.00	2.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	3.00	2.00

Continued on next page

Class	concept	DIT	LCOM	LOC	VG	MLOC	NBD	NOF	NOM	NORM	PAR	NSF	NSM	NSC	SIX	USEDBY	USES	WMC
org.tigris.scarab.-util.word.-IssueSearchTest	false	5.00	0.67	143.00	1.22	100.00	1.11	3.00	9.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	9.00	11.00
org.tigris.-scarab.util.word.-IssueSearchFactoryTest	false	5.00	1.50	172.00	2.67	91.00	2.00	1.00	3.00	0.00	0.33	1.00	0.00	0.00	0.00	0.00	16.00	8.00
org.tigris.-scarab.util.word.-IssueSearchFactoryTest$ISFactoryTest		1.00	0.67	63.00	1.75	38.00	1.50	8.00	4.00	0.00	1.50	0.00	0.00	0.00	0.00	2.00	16.00	7.00
org.tigris.scarab.-util.xmlissues.-ImportIssuesTest	false	5.00	1.50	20.00	1.00	6.00	1.00	0.00	2.00	0.00	0.00	2.00	0.00	0.00	0.00	1.00	5.00	2.00

APPENDIX D

Published Papers

D.1 ICPC 2007

Recovering Concepts from Source Code with Automated Concept Identification
As appeared in ICPC 07: Proceedings of the 15th IEEE International Conference on Program Comprehension

Recovering Concepts from Source Code with Automated Concept Identification

Maurice M. Carey
Dept. of Computer Science & Engineering
Arizona State University - Tempe
Box 878809, Tempe, AZ 85287-8809
mmcarey@asu.edu

Gerald C. Gannod*†
Dept. of Computer Science and Systems Analysis
Miami University
Oxford, OH 45056
gannodg@muohio.edu

Abstract

The complexity of the systems that software engineers build has continuously grown since the inception of the field. What has not changed is the engineers' mental capacity to operate on about seven distinct pieces of information at a time. Improvements like the widespread use of UML have led to more abstract software design activities, however the same cannot be said for reverse engineering activities. The well known concept assignment problem *is still being solved at the* line-by-line *level of analyzing source code. The introduction of abstraction to the problem will allow the engineer to move farther away from the details of the system, increasing his ability to see the role that domain level concepts play in the system. In this paper we present a technique that facilitates filtering of classes from existing systems at the source level based on their relationship to the core concepts in the domain. This approach can simplify the process of reverse engineering and design recovery, as well as other activities that require a mapping to domain level concepts.*

1. Introduction

The complexity of the systems that software engineers build has continuously grown since the inception of the field. What has not changed is the engineers' ability to deal with a limited number of datum at any given time, psychologists tell us that the average person has working memory capacity to operate on about seven distinct pieces of information at a time [1]. While improvements in the field like the widespread use of UML has led to more abstract software design activities, the same cannot be said for reverse engineering activities. The well known *concept assignment problem* is still being solved at the *line-by-line* level of analyzing source code. The introduction of abstraction to the problem would allow the engineer to move farther away from the details allowing his own limited resources to capture a broader picture of the system, increasing his ability to see the role that domain level concepts play in the system.

In this paper we present a technique that facilitates filtering of classes from existing systems at the source level based on their relationship to the core concepts in the domain. This allows a software engineer to work with a smaller subset of the system. The technique presented involves collecting object-oriented metric data from existing systems, which are then used in machine learning methods to create classifiers of the systems. The classifiers are then used to identify classes that are likely related to domain level concepts. As will be seen, we have gathered some interesting results that indicate this technique may ease the effort required to identify concepts in existing systems. This approach can simplify the process of reverse engineering and design recovery, as well as other activities that require a mapping to domain level concepts.

The remainder of this paper is organized as follows. Section 2 describes the context for our approach including information on the concept assignment problem, object-oriented metrics, and machine learning. Section 3 presents the process used to collect data from an existing system, as well as training a support vector machine. Section 4 shows an example of the filtering application of the approach to the Panda software. Section 5 evaluates the effectiveness of our approach by presenting the results obtained from analyzing example software systems. Section 6 discusses some areas of related work. Finally, Section 7 concludes and suggests future investigations.

2. Background

This paper makes use of ideas related to several areas of research. The concept assignment problem is similar to our goal of filtering the class diagram. Metrics are used as a way of assigning quantifiable attributes to a class. Finally, machine learning is used to automate the process of classification.

*This author supported by National Science Foundation CAREER grant No. CCR-0133956.
†Contact Author.

2.1 Concept Assignment Problem

Biggerstaff et al. [2] describes the concept assignment problem as recognizing concepts in software systems and building a model or human level understanding of the concepts. We see the concept assignment problem as a two part process. The first step is to identify concepts in the software system. The second step is to build a model of the concepts. This paper describes a new method of *identifying* concepts that are represented by classes, and using the identification to produce an abstraction of a recovered class diagram. Specifically, we describe an instance of the concept assignment problem dealing only with object-oriented classes. Object-oriented design suggests that we ignore the details of the implementation of a class, so we believe that analyzing object-oriented software at the class level is a valid approach to solving the concept assignment problem.

2.2 Object-Oriented Metrics

A metric is defined as a standard of measurement [3]. In this paper, we use metrics as quantifiable attributes of classes. We will be more interested in what a set of metrics might say about the class than what each individual metric implies. Since our focus is on attributes of classes we will be most interested in class level metrics.

Several object-oriented metrics are used in our approach. These metrics are primarily designed to capture information about the size and complexity of software at the class level. Figure 1 provides a brief summary of the metrics used in our work. These metrics were chosen because they were available to be analyzed. We have not optimized the metrics used, nor do we believe that is an appropriate step to take at this time because we do not want to over optimize the metrics used to the data set collected. The best way to view the role of the metrics collected here is as incomplete indicators of the concepts, each metric adds more information to the decision making process.

2.3 Machine Learning

This work makes use of two different machine learning algorithms developed by the machine learning community. The first, Support Vector Machines (SVM) are the primary algorithm we use to classify results. The second, k-nearest neighbors (KNN) is used to validate results obtained with our primary algorithm. We use both as a black-box that takes inputs that are n length vectors of real numbers and outputs classification decisions.

2.3.1 Support Vector Machines

Support Vector Machines (SVM) were first introduced by Boser et al. [4]. SVM are an example of a learning methodology known as supervised learning [5]. A learning methodology is an approach to creating programs that calculate answers by analyzing given examples while supervised learning uses examples consisting of input-output pairs. The goal in a learning problem is to find a function that maps the given inputs to the desired outputs.

SVM generate linear functions as their hypothesis. The hypothesis are then applied to attributes that have been transformed to a high dimensional feature space. The transformation of attributes to a feature space is carried out by the application of kernels to the example datum. An example of a kernel is the radial basis function kernel shown in Equation (3) [6], which intuitively describes a radius of

Number of Attributes (NOF) The total number of instance and static attributes in the class.

Number of Static Attributes (NSF) The number of class or static attributes in the class.

Number of Methods (NOM) The total number of instance and static methods in a class.

Number of Overridden Methods (NORM) The total number of methods in the class that override methods from an ancestor class.

Number of Static Methods (NSM) The number of static or class methods in the class. Henderson-Sellers refers to this as Number of Class Methods (NCM).

Number of Parameters (PAR) The number of parameters in a method. In this paper we use the average over the class.

Number of Subclasses (NSC) The total number of direct subclasses of this class, a.k.a Number of Children.

Method Lines of Code (MLOC) The number of lines of code contained in the body of all methods in the class.

Lack of Cohesion of Methods (LCOM) LCOM can be calculated as in (1) where $\mu(A_j)$ is the number of methods that access the attribute A_j, a is the number of attributes, and m is the number of methods. LCOM is a measure of the cohesiveness of a class where smaller values indicate more cohesive classes.

Nested Block Depth (NBD) The depth of nested blocks of code.

McCabe Cyclomatic Complexity (VG) The maximum number of independent circuits through the directed acyclic graph of a method. In this paper we use the average over the class.

Weighted Methods per Class (WMC) The sum of McCabe Cyclomatic Complexity for the n methods in the class i, calculated by the formula given in (2).

Depth of Inheritance Tree (DIT) The depth of the class in the inheritance hierarchy.

Specialization Index (SIX) Defined as $\frac{NORM \cdot DIT}{NOM}$. Designed so that higher values indicate classes that are more specialized.

$$\text{LCOM} = \frac{\left(\frac{1}{a}\sum_{j=1}^{a}\mu(A_j)\right) - m}{1 - m} \quad (1)$$

$$\text{WMC} = s_i = \sum_{j=1}^{n} V_{ij}(G) \quad (2)$$

Figure 1. Definition of Metrics Used

influence, where $i = 1, \ldots, \ell$, $j = 1, \ldots, \ell$, and σ is a parameter to the kernel that scales the radius of the support vectors influence.

$$K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = \exp\left(-\frac{||\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j||^2}{2\,\sigma^2}\right). \tag{3}$$

SVM used as binary classifiers must solve the optimization problem in Equation (4) [5]. Geometrically this translates into finding a linear separating hyperplane in the higher dimensional space. The hyperplane is optimized for maximal margin and defined by the *weight vector* $\vec{\mathbf{w}}$ and *bias* b. In actual practice the dual representation is maximized allowing optimization to take place in the kernel space with slack terms introduced along with a penalty parameter C. In this context, $\vec{\mathbf{x}}_i \in \mathbb{R}^n$ are the n-dimensional training vectors, $y_i \in \{1, -1\}$ are the classification labels for the training vectors, and ℓ is the number of vectors. The set of classification labels $\{1, -1\}$ correspond to classes that are concepts (1), and those that are implementation details (-1).

$$\begin{aligned} \min_{\vec{\mathbf{w}}, b} \quad & \langle \vec{\mathbf{w}} \cdot \vec{\mathbf{w}} \rangle \\ \text{subject to} \quad & y_i \left(\langle \vec{\mathbf{w}} \cdot \vec{\mathbf{x}}_i \rangle + b \right) \geq 1, \\ \text{where} \quad & i = 1, \ldots, \ell \end{aligned} \tag{4}$$

In our data set, $n = 14$ so the vector $\vec{\mathbf{x}}_i$ has 14 dimensions, each related to a metric. Examples of some sample vectors from the Panda data set are shown in Table 1. The metrics and their ordering are as in Table 1.

Table 1. Panda data set examples.

i	$\vec{\mathbf{x}}_i$	y_i
1	$< 0, 0, 3, 31, 11, 0, 0, 1.75, 4, 0.444, 2.75, 0.5, 0, 1 >$	-1
2	$< 4, 0, 2, 331, 92, 0, 2, 1.806, 32, 0.5, 2.556, 0.917, 0, 5 >$	1
3	$< 0, 0, 8, 51, 16, 0, 0, 1.071, 14, 0.862, 1.143, 0.929, 0, 1 >$	1
4	$< 1, 0, 2, 24, 10, 0, 2, 1.667, 2, 0.5, 3.333, 0, 0, 1 >$	-1

2.3.2 k-Nearest Neighbors.

k-Nearest Neighbors (KNN) [7] is the simplest instance-based machine learning method. The algorithm is based on all the training instances being points in an n-dimensional space $\mathbb{R}^n$. Instances are classified by calculating the Euclidian distance to all points in the training set. The Euclidian distance, shown in Equation 5, is calculated from $a_r(\vec{\mathbf{x}})$ (e.g., the rth attribute of the learning instance vector $\vec{\mathbf{x}}$).

$$d(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = \sqrt{\sum_{r=1}^{n} \left(a_r(\vec{\mathbf{x}}_i) - a_r(\vec{\mathbf{x}}_j)\right)^2} \tag{5}$$

The class of the k nearest points is then used as a simple majority rules vote, the class of the majority is assigned as the class of the point in question. An alternative is to weigh the class of the k nearest points based on the multiplicative inverse of their distance from the point in question.

3. Approach

This section describes our overall approach and presents our analysis of the implementation. Then we discuss the processes used to classify the systems, collect metrics, and generate the results.

3.1 Overview

The primary objective of our research is to study Hypothesis 1.

Hypothesis 1 *The vectors consisting of metrics measuring size and complexity of software components are strong indicators of classes that represent concepts within a domain.*

In other words, in a given domain the interesting concepts will be realized in software by classes or components whose size and complexity, measured by the metrics presented in Section 2.2, will be *recognizably* different from uninteresting classes that do not represent core concepts of the domain.

Figure 2 shows an overview of the process used in our approach. The *Data Collection and Classification* step is used to gather information about training data as well as data on the subject system. The result (*Metric and Classification Data* in Figure 2) is then analyzed in the *Data Analysis* step. The result of the *Data Analysis* step is a set of identified concepts that can be fed into any of a number of applications that use the concept identification to perform recovery of high level abstractions.

One of the intentions of our work is to support reverse engineering and design recovery by facilitating recovery of abstract models from as-built models. This approach builds a classifier that operates on a individual software systems, though future work may expand the scope to operate on multiple systems.

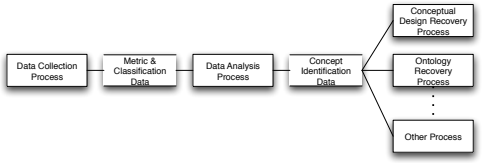

Figure 2. Approach Overview

3.2 Data Collection

The data collection process consists of collecting metrics data, transforming the data into a usable form by way of XML transformation, integrating manual classification data, and exporting the combined data to a format compatible with our analysis software.

The collection of metrics data begins with the import of the existing Java-based software system into the Eclipse platform and using the Eclipse plug-in Metrics [8] to generate the required metrics data. The metrics data is then

exported into an XML file for further processing.

An Excel XML spreadsheet file is then created from the XML-based metrics file using an XSL transform. This allows the data to be viewed or further manipulated in an easy to use environment. The spreadsheet file is integrated with the data collected from the manual classification step, resulting in a spreadsheet that contains all metric and classification data. Though we prefer the use of Matlab in our project for the classification and analysis of data it would also be possible to perform some analysis using a spreadsheet program. The final spreadsheet is then saved to a simple comma separated value (CSV) file. The CSV file is suitable for import into Matlab for analysis.

3.3 Manual Classification Step

The *manual classification* step is used to classify each component (class) from the system into one of two categories: *concept* or *other*. The purpose of this step is to provide a means to train and validate the classifier. The *concept* category is made up of components or classes that are representative of concepts from the knowledge domain of the system. The *other* category consists of everything that does not fit into the concept category. For example, consider a parser component. For a compiler this would be part of the concept category, since a parser is a core concept in this domain. In a word processor, this would likely not be classified to the concept category, since a parser would only support the primary or core concepts in the knowledge domain.

The approach to classification used in this paper involves examining the naming, relationships with other classes or concepts, and role within the architecture for the candidate class. This information must be evaluated with regard to the domain knowledge in order to classify the candidate class. It should be noted that there are no rules about how to classify a component. The decisions are based on the experience of the analyst, and their expertise in the domain. It is assumed that different analysts will produce slightly varying results. For our examples, we assumed any resulting classification that can be validated by a group of experts would be roughly equivalent to any classification we could arrive at or that the resulting error would not have a significant impact on the approach.

3.4 Machine Classification

The machine classification process consists of creating a SVM classifier from the data set then running the SVM classifier on the data set.

The data set is randomly broken up into two sets known as the *training set* and the *test set*. Both sets are of approximately equal size and consist of the metric data as well as the classification for all classes in the set.

The SVM classifier is created using the training set by an SVM algorithm implemented in Matlab. The metric data, in essence, is the domain of the classification function and the classification data is the range. The result is a set of support vectors that will be used to perform classification runs. The SVM algorithm takes a parameter, which in the case of the radial basis function is conceptually related to a radius of influence for a given support vector, that is selected by a process known as *cross-validation*. Cross-validation allows us to make an intelligent parameter selection by partitioning the training set into n partitions, this technique is designed to reduce the risk of overfitting. We then sample over a range of parameter options giving a set P of potential parameters. For each parameter in P we train n different classifiers by removing 1 of the n partitions from the training set for each classifier. We then evaluate the accuracy of the classifier on the remaining partition for each of the n different classifiers and compute an average accuracy for the classifiers with the given parameter value. The result is an optimal selection of the parameter from P based on how well the classifiers created with that value generalize to the n test partitions in addition to the accuracy they achieve over the partitioned training set. Once the parameter selection is finished we train a classifier with the given value over the complete training set.

The SVM classifier is used to analyze the metric component of the test data set, this step generates a predicted classification result. The predicted or observed classification is compared with the manual or expected classification derived manually from the class.

4. Example

In this section, we illustrate one of the potential uses of our approach. Specifically, we show how the concept identification technique can be used as part of a reverse engineering activity.

Panda [9] is a natural deduction proof assistant written in Java and developed at Arizona State University. The tool is used primarily to support education of natural deduction as a proof technique. We had access to the original class and statechart diagrams that were used for development of Panda and thus used that information as a sanity check during identification of concepts. Panda's size is 84 classes and about 9 thousand lines of code (KLOC).

The architecture of Panda is best described as a GUI based model view controller application. The model consists of classes implementing logical constructs. The view consists of classes implementing Java GUI elements. The controller consists of classes implementing the use cases that can be applied in a proof.

The basic process that we are using in this example is the following. First, we generate a class diagram for the system. Figure 3 shows a diagram for the Panda package in the Panda system. In addition, we generate the relevant metrics using the original source code for Panda. Second,

we use the SVM to identify those classes that are concepts in the system. Third and finally, we create an abstraction of the original class diagram that utilizes only those classes that were identified as concepts.

As shown in Figure 3, the Panda package has 45 classes. When we apply the SVM to the Panda data, 24 classes are filtered. When compared to a manual classification we performed, the automated classification produced 1 false positive and 4 false negatives. A more detailed evaluation of the approach is described in the next section.

Figure 4 shows the resulting class diagram using the information gathered using the SVM. As shown in the diagram, the resulting class structure is a more abstract representation of the original system. Specifically, it focuses more on domain concepts rather than details of the implementation. In this case, the remaining classes are related to logical formulas and the commands that operate upon them (e.g., the applicable inference rules). Filtered classes consisted of user interface classes and other implementation details.

5. Evaluation

Two software systems were used to evaluate our approach. In this section we first describe the systems analyzed and the statistical methods used to evaluate our results. Then we present the results of several experiments on data sets collected from two case study systems.

5.1 Description of Example Systems

Panda, as described earlier, is a natural deduction proof assistant. In the context of the concept identification work, Panda is used as a small system for illustration purposes.

Scarab [10] is a web-based defect tracking system based on the Turbine [11] and Torque [12] frameworks. Scarab was designed to be a replacement for defect tracking systems like Bugzilla [13]. The size of Scarab is 585 classes and 128 KLOC.

Turbine is a model view controller framework for web-based applications. Torque is an object-relational mapper for Java classes that allows objects to be persisted to the database. Scarab therefore has the architecture of a MVC web application whose model is persisted to a database.

5.2 Statistical Analysis Method

In order to evaluate the accuracy of the classifier, we performed a statistical analysis of the classification results. We calculated the *sample error* from the test set using Equation (6) [7] where n is the number of samples in the test set S, f is the function specified by our manual classification mapping the data point x to one of the two classes, h is the *hypothesis* generated by the learning algorithm, and the quantity $\delta(f(x), h(x))$ is defined in Equation (7). When the hypothesis disagrees with the manual classification, the sample error increases and $\delta(f(x), h(x))$ will be 1 for any instance x where the predicted classification $h(x)$ does not match the expected classification $f(x)$.

$$error_S(h) \equiv \frac{1}{n} \sum_{x \in S} \delta(f(x), h(x)) \tag{6}$$

$$\delta(f(x), h(x)) = \begin{cases} 1, & \text{if } f(x) \neq h(x) \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

We can estimate the *true error*, the error of the whole population, from the sample error using Equation (8) if we can meet the criteria in Figure 5. Here z_N is chosen based on the confidence level of the estimate.

$$error_{\mathcal{D}}(h) = error_S(h) \pm z_N \sqrt{\frac{error_S(h)(1 - error_S(h))}{n}} \tag{8}$$

1. $n \geq 30$,
2. the hypothesis commits r errors over the n samples,
3. the sample, or test set, S contains n data points which are drawn independently of each other,
4. the sample is chosen independently of the hypothesis, and
5. the sample is chosen according to the probability distribution $\mathcal{D}$ of the population.

Figure 5. Criteria for calculation of *true error*

Criteria 1 thru 4 of Figure 5 are met by the methods outlined in Section 3.4 and the size of the data set. Criteria 5 is more interesting since the dependency on the probability distribution requires that we test for a significant difference in the distributions of the test samples and the distribution of the population.

In order to confirm Criteria 5 we can perform the Kolmogorov-Smirnov hypothesis test on the population and sample distributions to test for significant difference. We formulate our hypothesis for this test in Hypothesis 2.

Hypothesis 2 *There is no significant difference in the distribution of the population and the sample data.*

The p-value of the Kolmogorov-Smirnov hypothesis test is the statistical significance, and α is the probability of rejecting the null hypothesis when it is in fact true. We will reject the null hypothesis if the p-value is less than α. Table 2 shows the results for each of the necessary test sets with an α value of 0.05 representing a confidence level of 95% for the test.

Using a *t*-test we have a toolset in place to detect situations where, for example, the accuracy is good, but there is no significant difference between the expected value (or

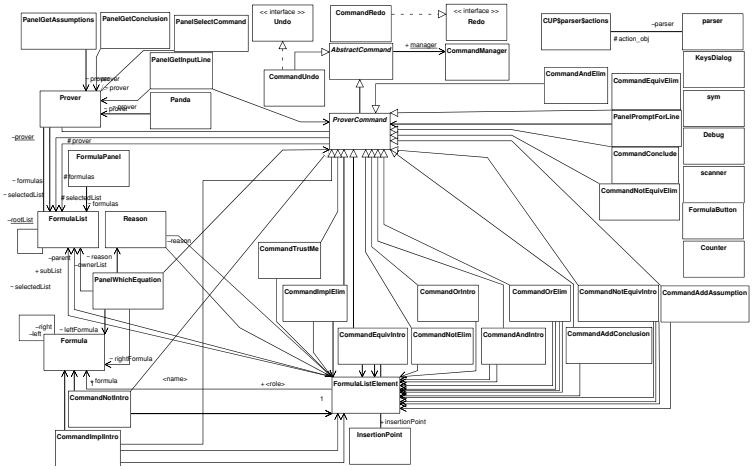

Figure 3. Panda class diagram

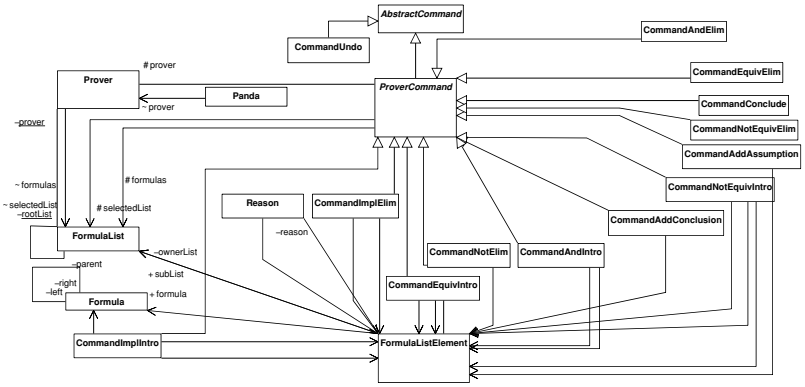

Figure 4. Panda class diagram after filtering

Table 2. Kolmogorov-Smirnov test $\alpha = 0.05$.

Data set	p-value	Reject Hypothesis 2
1 (Panda)	0.9922	No
2 (Scarab)	1.0000	No

mean) of the sample distribution and the measured accuracy. This could happen, as an example, in the case where the distribution of the expected results is 80% negative. The accuracy of the observed results could be 80% just by always guessing negative. The *t*-test will help to discover these situations by testing Hypothesis 3.

Hypothesis 3 *There is no significant difference in the mean of the sample data and the accuracy measured on that sample data.*

Rejecting Hypothesis 3 is a strong indicator that the accuracy is significantly different than the distribution of the data. Intuitively, a significant difference in the accuracy and the mean indicates that the classifier has learned something other than the expected value of the data set. As such, in Table 3 which shows the 95% confidence interval on the difference between population mean and sample accuracy, rejection of Hypothesis 3 is desirable.

5.3 Test Sets

Table 2 shows test sets 1 and 2 are randomly sampled from the Panda and Scarab data sets and are shown to be representative of the population they are sampled from by the acceptance of Hypothesis 2. Table 4 shows a mapping

Table 3. Results of Hypothesis 3 *t*-test.

Experiment	p-value	Reject Hypothesis 3
1 (Panda)	0.0198	Yes
2 (Scarab)	7.6288×10^{-11}	Yes
3 (Scarab KNN)	0.0753	No

of the test sets to the experiments they are used in.

Table 4. Test data mapping to experiments.

Experiment	Test set
1 (Panda)	1
2 (Scarab)	2
3 (Scarab KNN)	2

5.4 Experiments

This section shows the results of the seven experiments we have conducted thus far using the approach outlined in Section 3. The results are summarized in Table 5, which shows the sample accuracy along with predicted bounds of the *true accuracy* for N samples.

5.4.1 Experiment 1: Panda.

For this experiment we used the data set collected from Panda using the methods outlined in Sections 3.2 and Sections 3.3 . The process of classifying Panda took approximately four hours but given our previous experience with the software this may not be an indicator of the time required for an engineer looking at the system for the first time. We followed the process outlined in Section 3.4 to split the data into training and test sets, and used the training set to generate the SVM classifier used in this experiment. The purpose of this experiment was to attempt to identify evidence for the hypothesis using a fairly small system. We wanted to see if it was worth while to commit to the process for a larger system. The data set for Panda is under 100 elements and while a larger data set would better capture any generalizable properties of the hypothesis, this data set was simple to obtain and encouraged further analysis.

We used a SVM classifier based on a radial basis function kernel as in Equation (3) of Section 2.3.1 with the slack term of $C = 10$. Results are shown in Table 5. Accuracy was 80.43% for this sample. The *t*-test analysis (as indicated in Table 3) showed that the accuracy of the test versus the expected value of the results collected from the software were significantly different at the 95% confidence level, in other words the probability that the accuracy is different than the mean of the distribution is over 95%. These results were encouraging for a data set of this size, prompting us to proceed with the data collection effort for Scarab.

5.4.2 Experiment 2: Scarab.

For this experiment we used the data set collected from Scarab. Classification of Scarab required significantly more time than classifying Panda with over 40 hours of effort expended. This may have been a function of less familiarity with the system as well as the larger size. The process outlined in Section 3.4 was used to split the data into training and test sets, generate an SVM classifier for this experiment, and collect results. The purpose of this experiment was to validate the approach for a much larger data set than that used in the Panda experiment. Scarab is also a program that is more representative of systems in actual use.

The SVM classifier used was based on a radial basis function kernel with parameter of $C = 10$. Results are shown in Table 5. Accuracy for this sample set was 81.39%, and the *t*-test analysis showed that the accuracy of the test versus the expected value of the results were significantly different at the 95% confidence level. These are solid results that show quite a bit of promise for future research as it indicates that a fairly high degree of accuracy can be obtained using a moderate sized training set.

5.4.3 Experiment 3: Scarab Revisited.

For this experiment we wanted to repeat the Scarab experiment with a different machine learning algorithm for the classification. We chose the KNN classifier in order to form a simple baseline for comparison to the SVM results collected previously. The purpose of this experiment was to eliminate the chance that there is something special about the classification algorithm based on SVM.

In this experiment we used a value of $k = 3$. Results are shown in Table 5. The only parameter to the k-nearest neighbor algorithm is k. The sample accuracy is 76.92% which comes close to the accuracy obtained using SVM on the same data set in Section 5.4.2. The *t*-test analysis does not show a significant difference in the accuracy of the test verses the expected value as shown in Table 3. However, this result provides support for our hypothesis given that with 92% confidence we can show significant difference, this represents a 1 in 12.5 chance that the hypothesis is incorrectly rejected for this test. Given that we are over 99.999% confident in the results for experiment 2 and this test is only designed to confirm those results with a different machine learning algorithm it seems acceptable to view this test as a successful validation of those results even at only 90% confidence.

5.4.4 Discussion.

Table 3 gives us an indicator of how well the accuracy is measured. A small p-value is an indicator that there is a significant difference between the accuracy and the mean. Imagine a coin that is unfairly biased towards heads such that the expected value of heads is 80%. If we were to play a game where the coin is flipped repeatedly and the player must try to guess the result, the player would eventually discover the bias and begin guessing heads every round of the game. This learned strategy would lead to a measured accu-

Table 5. 99% confidence interval on accuracy of classification. (LB = lowerbound, UB = upperbound)

Experiment	Sample Accuracy	N	True Acc. LB	True Acc. UB
1 (Panda)	80.43%	46	68.97%	91.90%
2 (Scarab)	81.39%	403	77.59%	85.19%
3 (Scarab KNN)	76.92%	403	71.51%	82.34%

racy of the player at about 80%, in other words there would be no significant difference between the expected value of the experiment and the accuracy of the guess. In our results we believe that the machine is really making good predictions, because there *is* a statistically measurable difference between the expected value of the data set and the classification guess of the machine for all of the SVM predictions. So the results show support for Hypothesis 1.

These results lead to the idea that a SVM classifier trained within an application will work well on that application. This result has practical applications since a software engineer trying to identify concepts within a system with a large number of classes could start by classifying parts of the system. The approach described could be used to predict results for the remainder of the system. As the software engineer accepts or corrects the predictions they become more accurate. So the approach can be used as an interactive concept identification assistant that becomes more accurate as the system is classified. Those familiar with SPAM filters such as SpamAssassin [14] will recognize this concept, the more you train it the less time you spend dealing with SPAM. In presentation of this work to peers many practicing engineers appreciate the potential time savings. Their work with systems that have a large number of classes shows the immediate practical benefits to an approach that trains a computer to do a repetitive classification task.

5.5 Threats to Validity

The threats to validity include internal and external factors. The internal factors include errors in the statistical analysis, and overfitting of the data set. The external factors include manual classification errors, and inaccurate or poor definition of the concepts in the domain. We will describe each in further detail below.

The primary internal threat to validity that should be addressed is the difficulty in deciding what constitutes a *good result*. The difficulty stems from the distribution of classifications where there are more negatives than positives. In other words there are fewer classes representing concepts than implementation. It has been shown in Section 5.2 that the accuracy of the prediction can not be used in isolation, but must be considered along with the *t*-test analysis of the distributions' expected value verses the accuracy. This gives a more complete picture of the result showing some indication as to how significant the accuracy is. The assumption here was that results that had better than average accuracy along with a significant difference in mean were good results that support our hypothesis.

Overfitting is a threat to validity that effects any machine learning approach, but is only a secondary threat to the validity of our approach. Overfitting means that results will not generalize well to other data sets, since the machine has learned a pattern for a specific data set. In other words if the SVM is overfitted to the data then we could not expect good results from a different data set classified with the same machine. At this point we are not that concerned about generalization to other data sets as we have a practical application of the approach in the classification of a single large system. In future work the potential of overfitting will play a larger role in the threat to internal validity of our approach if it is expected that our approach will produce general results.

The primary external threat to validity is misclassification during the manual classification process. There are two possible scenarios that result in the introduction of inaccurate analysis into the system and they are either a logical error or a typographic error on the part of the engineer. We have worked with enough software engineers to know that each has a different opinion on any given subject, these differing opinions would appear in the system as logical errors. However, this assumes that one of the engineers is more correct than the other within some externally specified system of objective truth that likely does not exist, and this ignores the possibility of having more than one correct representation of the system. We must concede that any accurate automated classification is only as accurate as the engineer who trained it, but this really is not problematic as long as the engineer is consistent and in fact is representative of the results of a manual classification. The typographic errors that could have been introduced into our data set were minimized by careful data entry, along with rechecking each result against the class. Though we believe we have minimized the errors as much as possible we have no way to measure this using the current process, and as is the case with many machine learning applications we can not predict the effect of a classification error on the resulting classifier.

A secondary external threat to validity is the design of the software itself. If poorly designed software systems are introduced it is difficult to predict the results. As an example imagine a single concept being represented by two classes. There may be instances where the lack of cohesion makes sense but it may not be possible to decide where.

6. Related Work

Biggerstaff et al. [2] describe the assignment of human oriented concepts to implementation oriented concepts. This process is a matching between preexisting notions about the system or existing domain knowledge to textual representations of the system implementation. While the definition of the concept assignment problem given fits well with the filtering application we present here there are a few key differences. The approach used by Biggerstaff et al. uses source code where we use metrics based on the source code. We use a machine learning approach that does not involve declaratively creating an expert system. They make a strong case for the methods used to recognize concepts, using at times a filtering of the source code. Our approach could be added to the toolset mentioned here as an additional filtering technique, though in time we will automate more of the process to the point where interaction with the user is very limited.

Svetinovic et al. [15] discuss concept identification from a forward engineering perspective illustrated in several case studies. The claims of misuse or misunderstanding of object-oriented domain analysis are worth noting since the automated identification of concepts requires that those concepts are represented within the implementation, implying they where designed into the software. They identify many of the concerns we have raised in our discussion of external validity. Primarily, that there is not one single agreed upon way of designating what is and is not a concept in a software system, according to the paper this is a fundamental difficulty with object-oriented domain analysis.

Merlo et al. [16] describe a design recovery process which uses a machine learning approach to link text from source code to concepts identified by a domain analysis. This is essentially a partial automation of a portion of the work performed by Biggerstaff et. al. The machine learning algorithm used is based on neural networks. The approach differs from ours in that a change in domain requires a new neural net to be trained. That is not necessarily the case with our SVMs as they can be applied to programs from very different domains but currently without significant accuracy.

Hsi et al. [17] approach recovery of ontology by manual construction of an *interface map*. The interface map is a depth first traversal of all elements of the softwares interface. The approach then uses the interface map to generate a semantic network. The semantic network is then used as the basis for calculating graph theoretic measures that are used as indicators of *core concepts* of the ontology. The approach differs from our work in that it is based on construction and analysis of a graph. The evaluation does not include any comparison to a *control group* in order to express the accuracy of the approach. Each of the varying methods introduced to recover the ontology produce differing ontologies, and no method of reconciling differing results is presented. Advantages of our approach include the use of a control group in the form of the manual classification results to show the accuracy of the approach, and we only produce one ontology based on the metrics that we use. This paper presents an interesting set of metrics that may add to our approach in future work.

Software architecture reconstruction is a recent development that represents a reverse engineering process designed to recover architecture specific information. As Favre [18] points out, software architecture is not well defined. Software architecture can only be defined in terms of the audience to which it is presented. An experienced software engineer will recognize the overloaded use of the term to have slightly differing meanings based on who is the target audience. Obviously, the project manager, technical lead, upper management, customers, fellow engineers with similar experience, the newest addition to the team who finished his degree only a few months ago, as well as other stakeholders all have a different perspective on the software architecture. van Deursen et al. [19] seem to ignore this ambiguity but the approach of view-driven software architecture reconstruction called Symphony is introduced. This paper essentially codifies best practices that have been observed by the authors in the actual practice of software architecture reconstruction. Placed in the context of software architecture reconstruction and in the vocabulary of Symphony our work would be a type of mapping from a source view to a target view, the target view being a static graph of the core concept classes described by a UML class diagram.

Marcus et al. [20] describe a method of using latent semantic indexing to map natural language concepts to the locations of those concepts in source code. Zaidman et al. [21] show that webmining techniques can be applied to source code to uncover important classes, similar in nature to our core concept classes.

7. Conclusion and Future Work

The results strongly indicate that there is a relationship between the given metrics and the identification of concepts. The relationship that is captured by the SVM classifier produces above average results over a single software system. However, a larger data set is needed to comprehensively verify these results. Additionally, we have made no attempt to optimize the metrics that were selected for this set of experiments. In fact we initially simply selected all metrics available from the output of the metrics software used thinking that we would have to optimize the feature vector in order to obtain acceptable results. Obviously this is an area of future research since optimal results may allow a stronger formulation of our hypothesis, particularly in regards to what type of metrics are the best indicators.

The example filtering shown in Section 4 effectively demonstrates one of the practical applications of the clas-

sification technique presented here. The class diagrams depicted in Figure 3 and Figure 4 show a visual demonstration of the effectiveness of the filtering process. Figure 4 is simply easier to understand due to the smaller number of classes. The effect is even more obvious and useful on systems with many more classes.

Work on this project is ongoing. We currently have plans to extend our data sets significantly. This involves classifying several different systems from ideally orthogonal domains. We are also investigating other possible applications of this technique. Plans are underway to develop an Eclipse plugin to allow this technique to integrate with the Eclipse [22] environment, providing online classification suggestions to the developer. This represents an exciting practical application of the results even lacking generalization to more than one system, which is another area of future interest. We are also interested in increasing the accuracy of the classifier. This involves developing a deeper understanding of the relationship that the classifier is detecting. We will be investigating the use of different kernels as well as performing more analysis of the features to determine what metrics are the best indicators. We will explore the learning curve of the classifier, how quickly can it be trained to acceptable accuracy. Additionally we are interested in training optimizations, some training sets are better than others. Is it possible to select them prior to manual classification.

References

[1] George A. Miller. The magical number seven, plus or minus two. *The Psychological Review*, 63:81–97, 1956.

[2] Ted J. Biggerstaff, Bharat G. Mitbander, and Dallas E. Webster. Program understanding and the concept assignment problem. *Communications of the ACM*, 37(5):72–82, 1994.

[3] Brian Henderson-Sellers. *Object-Oriented Metrics: Measures of Complexity*. Object-oriented series. Prentice Hall PTR, 1996.

[4] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM Press.

[5] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[6] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2002.

[7] Tom M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

[8] Frank Sauer. Metrics 1.3.6. [Online] Available http://metrics.sourceforge.net/.

[9] Greg Hodgdon. PANDA : Proof Assistant for Natural Deduction Analysis. Technical Report of MCS Project, Arizona State University, November 2001.

[10] Tigris. Scarab. [Online] Available http://scarab.tigris.org/.

[11] Apache Organization. Turbine. [Online] Available http://jakarta.apache.org/turbine/.

[12] Apache Organization. Torque. [Online] Available http://db.apache.org/torque/.

[13] Bugzilla Organization. bugzilla.org. [Online] Available http://www.bugzilla.org/.

[14] Apache Organization. The apache spamassassin project. [Online] Available http://spamassassin.apache.org/.

[15] Davor Svetinovic, Daniel M. Berry, and Michael Godfrey. Concept identification in object-oriented domain analysis: Why some students just don"t get it. In *RE '05: Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 189–198, Washington, DC, USA, 2005. IEEE Computer Society.

[16] Ettore Merlo, Ian McAdam, and Renato De Mori. Feed-forward and recurrent neural networks for source code informal information analysis. *Journal of Software Maintenance*, 15(4):205–244, 2003.

[17] I. Hsi, C. Potts, and M. Moore. Ontological excavation: unearthing the core concepts of the application. In *WCRE 2003: Proceedings of 10th Working Conference on Reverse Engineering*, pages 345–353, 2003.

[18] J. M. Favre. Cacophony: metamodel-driven software architecture reconstruction. In *WCRE 2004: Proceedings of the 11th Working Conference on Reverse Engineering*, pages 204–213, 2004.

[19] A. van Deursen, C. Hofmeister, R. Koschke, L. Moonen, and C. Riva. Symphony: view-driven software architecture reconstruction. In *WICSA 2004: Proceedings of Fourth Working IEEE/IFIP Conference on Software Architecture*, pages 122–132, 2004.

[20] Andrian Marcus, Andrey Sergeyev, Vaclav Rajlich, and Jonathan I. Maletic. An information retrieval approach to concept location in source code. In *WCRE '04: Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04)*, pages 214–223, Washington, DC, USA, 2004. IEEE Computer Society.

[21] Andy Zaidman, Toon Calders, Serge Demeyer, and Jan Paredaens. Applying webmining techniques to execution traces to support the program comprehension process. In *CSMR '05: Proceedings of the Ninth European Conference on Software Maintenance and Reengineering*, pages 134–142, Washington, DC, USA, 2005. IEEE Computer Society.

[22] Eclipse Organization. Eclipse. [Online] Available http://www.eclipse.org.