Defining a Software Analysis Framework

Oguzhan Dogan

Aug
2008

MSI
Växjö University
SE-351 95 VÄXJÖ

Report 08079
ISSN 1650-2647
ISRN VXU/MSI/DA/E/--08079/--SE
Acknowledgement

Thanks to…
... Professor Welf Löwe and other developers of VizzAnalyzer,
... Mathias Hedenborg for his lectures,
... Rudiger Lincke for his lectures and providing me with inspiring notes and interesting new ideas for my project,
... Jesper Andersson for his lectures,
... My good class mates who attended the same course with me.
Växjö, May 2008
Abstract

Nowadays, assessing software quality and making predictions about the software are not possible. Software metrics are useful tools for assessing software quality and for making predictions. But currently the interpretation of the measured values is based on personal experience. In order to be able to assess software quality, quantitative data has to be obtained.

VizzAnalyzer is a program for analyzing open source Java Projects. It can be used for collecting quantitative data for defining thresholds that can support the interpretation of the measurement values. It helps to assess software quality by calculating over 20 different software metrics. I define a process for obtaining, storing and maintaining software projects. I have used the defined process to analyze 60-80 software projects delivering a large database with quantitative data.

**Keywords:** SourceForge.NET, Software Metrics, VizzAnalyzer.
# Table of Contents

1. Introduction........................................................................................................................................... 1  
   1.1. Context of thesis.......................................................................................................................... 1  
   1.2. Problem Description...................................................................................................................... 1  
   1.3. Goals and Criteria......................................................................................................................... 1  
   1.4. Motivation...................................................................................................................................... 2  
   1.5. Outline........................................................................................................................................ 2  
2. Background............................................................................................................................................. 3  
   2.1. Java Programming Language........................................................................................................ 3  
   2.2. What is Eclipse?............................................................................................................................. 4  
   2.3. VizzAnalyzer and Software Metrics.............................................................................................. 5  
   2.3.1. Software Metrics...................................................................................................................... 6  
   2.4. What is SourceForge.NET?........................................................................................................... 9  
   2.5. Related Works................................................................................................................................ 10  
3. Description of the Collected Data........................................................................................................ 11  
4. Process Description............................................................................................................................... 14  
   4.1. Downloading Software Projects................................................................................................... 14  
   4.2. Compiling and Analyzing Software Projects............................................................................... 15  
   4.3. Documentation and Figure of the Process Flow......................................................................... 17  
5. Analysis Result....................................................................................................................................... 20  
   5.1. Collected Projects......................................................................................................................... 20  
   5.2. Analysis result and Problems....................................................................................................... 21  
6. Conclusion and Future Work................................................................................................................ 24  
   6.1. Conclusion..................................................................................................................................... 24  
   6.2. Future Works............................................................................................................................... 24  
Bibliography.............................................................................................................................................. 25  
Appendix A............................................................................................................................................... 26  
   1. Installation of Eclipse Platform and VizzAnalyzer......................................................................... 26  
   2. Statistic Figures............................................................................................................................... 28  
   3. Collected Projects............................................................................................................................. 30
Index of Tables
Table 2.1: Software Metrics (1) ........................................................................................................8
Table A.1: Collected Java Projects (13) ..........................................................................................37
Index of Figures

Figure 2.1: Screenshot of Eclipse ............................................................... 5
Figure 2.2: Quick Fix (9) ........................................................................... 5
Figure 2.3: VizzAnalyzer Framework .......................................................... 6
Figure 2.4: Screenshot of SourceForge.net (12) ......................................... 9
Figure 3.1: Database Environment ............................................................. 11
Figure 3.2: Intended Audience ................................................................. 12
Figure 3.3: Operating Systems ................................................................. 12
Figure 3.4: Topic ....................................................................................... 13
Figure 3.5: User Interface ........................................................................ 13
Figure 4.1: A software project web page .................................................. 14
Figure 4.2: File Releases .......................................................................... 15
Figure 4.3: CVS and SVS ......................................................................... 15
Figure 4.4: Problems View ....................................................................... 16
Figure 4.5: VizzAnalyzer ......................................................................... 16
Figure 4.6: Selection of UNIX name for analyzing ...................................... 17
Figure 4.7: Typing Project Detail ............................................................... 17
Figure 4.8: Project Details ....................................................................... 18
Figure 4.9: Process Flow .......................................................................... 19
Figure 5.1: Screenshot of workspace ......................................................... 20
Figure 5.2: Screenshot of the folders related with projects ......................... 21
Figure 5.3: Problem Statistic .................................................................... 22
Figure A1.1: Eclipse Folder ...................................................................... 26
Figure A1.2: Workspace Launcher ............................................................ 27
Figure A1.3: Eclipse Welcome Page .......................................................... 27
Figure A1.4: Screenshot of Eclipse ............................................................. 28
Figure A2.1: Number of Projects with problem and without problem ........ 28
Figure A2.2: Projects and Problems for each years .................................... 29
Figure A2.3: Problem types and numbers .................................................. 29
Figure A2.4: Missing Libraries ................................................................. 30
1. Introduction

This chapter gives description of problem solved by this thesis project and the goals-criteria need to be met. Furthermore, it describes the motivation to do this thesis project and how it is achieved.

1.1. Context of thesis

Nowadays, the interpretation of the measured values is based on personal experiences. Quantitative data has to be obtained for allowing statistical evaluations and further investigations since personal experiences is not enough for allowing statistical evaluations and further investigations.

“VizzAnalyzer is a quality analysis tool. It reads software codes and other design specifications and documentations and performs a number of quality analyses. It is easily adaptable to the quality needs of a certain company or project and can be harmoniously integrated in existing development processes.” (1)

Currently, a prerequisite exists, which is that the VizzAnalyzer requires that the project compile without error for being able to analyze it. To be able to obtain the statistical data without problems, a significant amount of projects being ready to be analyzed is needed. Further, the process for obtaining these software projects should be as automated as possible, repeatable, transparent, extendibles and flexible.

1.2. Problem Description

Qualitative values explore attitudes, behavior and experiences. Qualitative values provide to us general opinion from participants. Quantitative values are statistic measured information. Quantitative researches are more reliable than qualitative researches. I can give an example about qualitative and quantitative measures like: to calculate the weight of a person by carrying her is qualitative measure and to calculate it by a weightlifter is quantitative measure.

In software technology, quantitative values are software metrics. They are so important to assess software quality but currently we can just get qualitative value for software projects. Software metrics have to be obtained for creating statistical evaluations and further investigations.

VizzAnalyzer is a useful software analysis tool to obtain software metrics from the Java projects by analyzing them. VizzAnalyzer calculates over 20 software metrics which we can use to assess software quality.

The problem addressed by this thesis is the definition of a process for obtaining, storing and maintaining a large number of open software projects, to create a repository containing a large number of projects and to prepare them for analysis with the VizzAnalyzer and to extract software metrics. The data shall be provided for later analysis. The process needs to be automated as far as possible, repeatable, transparent, extendibles and flexible. This problem is not trivial to solve, since some open source projects are not compiling when they are downloaded, and thus are not ready for being analyzed by the VizzAnalyzer.

1.3. Goals and Criteria

This part describes the goals pursued by this thesis in order to solve problem and the criteria user for validating the goals:

- The first goal is to get good understanding Eclipse and Java programming language for making Java projects suitable for VizzAnalyzer. The important
part of this goal is to be able to solve the problems while compiling projects. This goal is reached if Java projects compile without error. This goal is fulfilled when I compile Java projects and analyze them.

- The second goal is to get knowledge about SourceForge.net for downloading open source projects. This goal is fulfilled when I get Java projects to my workspace. By achieving this goal I will be able to obtain Java projects.
- The third goal is to analyze Java projects by using VizzAnalyzer. VizzAnalyzer allows us to get quantitative values from software projects. This goal is fulfilled when 60-80 Java projects are analyzed and I get quantitative values and database files which include software projects properties.
- Finally, summarizes of all goals and criteria: the goal is to define a process for getting, storing and maintaining the Java projects, to create a repository containing those Java projects and to prepare them for analyzing with VizzAnalyzer and to get software metric from VizzAnalyzer. This entire goal is fulfilled when 60-80 Java projects are being ready for VizzAnalyzer and analyzed by VizzAnalyzer.

1.4. Motivation

One of the most importance concepts in science is measurement and analysis. Measurement and analysis are used for gathering quantitative data. Quantitative values help developers to manage projects. Software metrics are quantitative values for the software projects in computer science.

“Software metrics are useful data for assessing software quality and for making predictions. But currently the interpretation of the measured values is based on personal experience and gut feeling. Not much information is available about thresholds and possible ranges of the metric values. In order to be able to define thresholds on which general recommendations could be based, quantitative data has to be obtained for allowing statistical evaluations and further investigations.” (2)

It is worthwhile to prepare analyzable projects for VizzAnalyzer and define a process for analyzing them since currently no larger number of the open source software projects exist which could be easily analyzed for obtaining software metrics.

1.5. Outline

The remainder of this thesis is structured as following:

Chapter 2 presents the background knowledge necessary for understand certain parts of this thesis. It discusses the VizzAnalyzer, Eclipse and SourceForge.net. Chapter 3 describes the requirements. It includes the projects descriptions as subsection. Chapter 4 covers the process flow for my project. It presents all of the steps for achieving my project. Chapter 5 presents the analyze results of Java projects. All drafts description will be stated at this chapter. Finally, Chapter 6 will discuss conclusion and future work.
2. Background
This chapter describes the technologies that are related to the problem domain. It gives a brief description about the Java programming language, Eclipse platform, VizzAnalyzer, SourceForge.net and the software metrics.

2.1. Java Programming Language
Java is an object oriented programming language developed by Sun Microsystems. Java provides a lot of advantages for programmers. In this programming language: (3)
• It is easy to code program because it uses object oriented methodology.
• In this programming language, development is much faster because Java technology is interpreted.
• Applications are portable across multiple platforms. With Java, write applications once, and never need to port them. Applications will run without modification on operating systems and hardware architectures.
• Applications are robust because the Java runtime environment manages memory for developer.
• Interactive applications have high performance.
• Applications are adaptable to changing environments because code modules can be dynamically downloaded from anywhere on the network.
• End users can trust that your applications are secure, even though they're downloading code from all over the Internet, because; the Java runtime environment has built-in protection against viruses and tampering.

The following paragraphs summarize some features of Java particularly relevant for this project. Class Path, Java Libraries and syntax errors are three of them.

The Class path is an argument set on the command-line, or through an environment variable, that tells the Java Virtual Machine where to look for user-defined classes and packages in Java programs. (4)

Contrary to the classic dynamic loading behavior, when executing Java programs, the Java Virtual Machine finds and loads classes lazily. The byte code of a class is loaded only when this class is first used. The virtual machine searches and loads classes in this order: (4)
• Bootstrap classes: the classes that is fundamental to the Java Platform (comprising the public classes of the Java Class Library, and the private classes that are necessary for this library to be functional).
• Extension classes: packages that are in the extension directory of the JRE or JDK.
• User-defined packages and libraries.

By default only the packages of the Java Development Kit standard Application Programming Interfaces, and extension packages are accessible without needing to set where to find them. The path for all user-defined packages and libraries must be set in the command-line (or in the Manifest associated with the Jar file containing the classes).

Java programs consist of classes and objects. Classes consist of methods. Methods perform tasks and return information at the end of their tasks. When a programmer wants to code a program with Java, programmer can program each Java element. However, most Java programmers take advantage of rich collections of existing classes in Java class libraries. The Java class libraries are also known as the Java Application Programming Interfaces. Class libraries are provided primarily by compiler vendors, but many class libraries are supplied by independent software vendors. Also, many class
libraries are available on Internet. Everybody can download free ware products and use them for free. (5)

Java class libraries serve three purposes within the Java platform: (6)

• Like other standard code libraries, they provide the programmer a well-known set of functions to perform common tasks, such as maintaining lists of items or performing complex string parsing.

• In addition, the class libraries provide an abstract interface to tasks that would normally depend heavily on the hardware and operating system. Tasks such as network access and file access are often heavily dependent on the native capabilities of the platform.

• Finally, some underlying platforms may not support all of the features a Java application expects. In these cases, the class libraries can either emulate those features using whatever is available, or provide a consistent way to check for the presence of a specific feature.

While programmer programs, it is possible to forget to write some or write it wrong. When it is happened, Java gives syntax error. More information about Java programming language could be found in “Java-A Beginner's Guide” (7) and also on Internet.

2.2. What is Eclipse?

The first tool that we need is a platform which allows us to develop Java projects and use VizzAnalyzer on it. Eclipse is the common tool which used for developing Java projects all over the world. It is an open source community.

Eclipse is a development environment written in Java. In its default form it is meant for Java developers, consisting of the Java Development Tools. At the core of the Eclipse is architecture for dynamic discovery, loading and running of plug-in. We can extend its capabilities by installing plug-in written for Eclipse software framework such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. (8) Each plug-in can focus on doing a small number of tasks well or they can work together for a task. VizzAnalyzer is one of good examples for this combination.

Eclipse provides plug-in to run on it. After launching Eclipse, to use “Help” menu and “Software Update” sub menu and instructions will help us to install new plug-in on Eclipse. We can see a screen shot of Eclipse in Figure 2.1. On the left side of this figure, we can see all projects which we are working on it, on the right side there are some tools help us to change some properties of projects and on the bottom we can find the description of the problem. we can find more information about how to install and launch eclipse in Appendix A and on Internet.
Eclipse includes some features for projects like; Syntax highlighting, content/code assist, code formatting, import assistance, quick fix and integrated debugging features. Eclipse offers corrections to problems found while typing and after compiling. To show that correction proposals are available for a problem or warning. User can select suitable solution by clicking to warning. Figure 2.2 shows an example for this case. On the left side there is a warning and also we can see suggestion for this case. We can use Eclipse “Help” menu to find more information about Eclipse and those features.

2.3. VizzAnalyzer and Software Metrics
This part describes VizzAnalyzer and software metrics which are calculated by VizzAnalyzer.

“VizzAnalyzer is a quality analysis tool. It reads software codes and other design specifications and documentations and performs a number of quality analyses. It calculates over 20 software metrics. It is easily adaptable to the quality needs of a
certain company or project and can be harmoniously integrated in existing development processes.” (1)

“The VizzAnalyzer is a reusable framework extensible for new programming languages, analyses, and visualization tools. This external plug-ins is connected to the framework by wrappers. They are responsible for handling VizzAnalyzer and adapting different data formats. For standard formats, a number of predefined converters are available.” (1) Figure 2.3 shows VizzAnalyzer Framework. It includes converter, wrapper and plug-in parts as we can see in this figure.

More information about VizzAnalyzer is available on Internet. (1)

2.3.1. Software Metrics

Software metric is a measure of some property of a piece of software or its specifications. Since quantitative methods have proven so powerful in the other sciences, computer science practitioners and theoreticians have worked hard to bring similar approaches to software development. Tom DeMarco stated, “You can’t control what you can't measure.” (10) Software metrics tend to be used as an aid in judging the quality of software projects.

The Metrics analysis measures certain properties in the graph they are applied on and abstract these measures in numerical values, which can be related to quality property of a software system.

A list of software metrics and description could be seen in Table 2.1; software metrics names are shown with italic font style and than we can see the description of those 21 metric. More information about software metrics is available on VizzAnalyzer web site. (1)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1-Lines of Code (LOC): Lines of code simply count the lines of source code (line break characters) of a certain software entity. It is a simple yet powerful metric to assess the complexity of software entities. Since it is depending on code conventions and format, it is critical to use it in generated codes since it may lack of line breaks. Additionally it can only be measured in the source code itself from
the front-end and is therefore a front-end side metric.

2-Number of Attributes and Methods (SIZE2): Number of Attributes and Methods simply counts the number of attributes and methods of a class. It is an object-oriented metric that can be applied to modular languages by considering the number of variables (globally visible in a module) and its number of functions and procedures.

3-Number of local Methods (NOM): Number of local Methods measures the number of methods locally declared in a class. Inherited methods are not considered. It is the size of the interface of a class and allows conclusions on its complexity.

4-McCabe Cyclomatic Complexity (CC): CC is a measure of the control structure complexity of software. It is the number of linearly independent paths and therefore, the minimum number of independent paths when executing the software.

5-Weighted Method Count (WMC): A weighted sum of methods implemented within a class. It is parameterized by a way to compute the weight of each method. Possible weight metrics are:
   - McCabe Cyclomatic Complexity,
   - Lines of Code,
   - 1 (unweighted WMC).

This variant of WMC uses McCabe Cyclomatic Complexity metric for calculating the weight for each method. Originally defined as an object-oriented metric, it can easily adapt to non-object-oriented systems computing the weighted sum of functions implemented within a module or file.

6-Afferent Coupling (Ca): Afferent coupling between packages (Ca) measures the total number of external classes coupled to classes of a package due to incoming coupling (coupling from classes external classes of the package, uses CBO definition of coupling). Each class counts only once. Zero if the package does not contain any classes or if external classes do not use the package's classes. Ca is primarily applicable to object-oriented systems.

7-Response For a Class (RFC): Count of (public) methods in a class and methods directly called by these. RFC is only applicable to object-oriented systems.

8-Depth of Inheritance Tree (DIT): Depth of Inheritance Tree (DIT) is the maximum length of a path from a class to a root class in the inheritance structure of a system. DIT measures how many super-classes can affect a class. DIT is only applicable to object-oriented systems.

9-Number of Children (NOC): NOC is the number of immediate subclasses (children) subordinated to a class (parent) in the class hierarchy. NOC measures how many classes inherit directly methods or fields from a super-class. NOC is only applicable to object-oriented systems.

10-Coupling Between Objects (CBO): Coupling between Objects (CBO) is the number of other classes that a class is coupled to. CBO is only applicable to object-oriented systems.

11-Change Dependency between Classes (CDBC): The Change Dependency between Classes (CDBC) measures the class level coupling. It is a measure assigned to pairs of classes describing how dependent one class (client class) is on the other (server class). This allows conclusions on the follow-up work to be done in a client class, when the server class is changed in the course of re-engineering.

12-Efferent Coupling (Ce): Efferent Coupling between packages (Ce) measures the total number of external classes coupled to classes of a package due to
outgoing coupling (coupling to classes external classes of the package, uses Ce definition of coupling). Each class counts only once. Zero if the package does not contain any classes or if external classes are not used by the package's classes. Ce is primarily applicable to object-oriented systems.

13-Coupling Factor (CF): Coupling Factor (CF) measures the coupling between classes excluding coupling due to inheritance. It is the ratio between the number of actually coupled pairs of classes in a scope (e.g., package) and the possible number of coupled pairs of classes. CF is primarily applicable to object-oriented systems.

14-Instability (I): Instability between packages (I) measures the ratio between the outgoing and the total number of in- and outgoing couplings from classes inside the package from/to classes outside the package (coupling to classes external classes of the package, uses I definition of coupling). Each class counts only once. Zero if the package does not contain any classes or if external classes are not used by the package's classes. It is primarily applicable to object-oriented systems.

15-Locality of Data (LD): The Locality of Data metric relates the amount of data being local the class to the total amount of data used by the class. This relates to the quality of abstraction embodied by the class and allows conclusions on the reuse potential of the class and testability.

16-Message Passing Coupling (MPC): The MPC measures the number of method calls defined in methods of a class to methods in other classes, and therefore the dependency of local methods to methods implemented by other classes. It allows for conclusions on the message passing (method calls) between objects of the involved classes. This allows for conclusions on re-usability, maintenance and testing effort.

17-Package Data Abstraction Coupling (PDAC): The PDAC measures the coupling complexity caused by ADTs on package level. Based on DAC it transfers the effects of the coupling between classes on the reuse degree, maintenance and testing effort to more abstract organization units like packages and modules, which are as well decisively influenced by the coupling between classes of different packages.

18-Lack of Cohesion in Methods (LCOM): The Lack of Cohesion in Methods metric is a measure for the number of not connected method pairs in a class representing independent parts having no cohesion. It represents the difference between the number of method pairs not having instance variables in common, and the number of method pairs having common instance variables.

19-Improvement of LCOM (ILCOM): The Improvement of LCOM (cf. Lack of Cohesion in Methods) metric is a measure for the number of connected components in a class. A component are methods of a class sharing (being connected by) instance variables of the class. The less separate components there are the higher is the cohesion of the methods in the class.

20-Tight Class Cohesion (TCC): The Tight Class Cohesion metric measures the cohesion between the public methods of a class. That is the relative number of directly connected public methods in the class. Classes having a low cohesion indicate errors in the design.

21-Lack of Documentation (LOD): How many comments are lacking in a class, considering one class comment and a comment per method as optimum. Structure and content of the comments are ignored.

Table 2.1: Software Metrics (1)
2.4. What is SourceForge.NET?
This part summarizes the SourceForge.net. SourceForge.net is the world’s largest open source software development web site. SourceForge.net has the largest repository of open source code and applications available on the Internet and hosts more open source development products than any other site. (11)

The screenshot of the SourceForge.net could be seen in Figure 2.4; shows the main page of SourceForge.NET, includes “Search” is tool for searching, “Project News” is for getting the last news about the projects and “Browse Software” is for general looking to the projects. We can also create a project by clicking “Create Project”.

![SourceForge.net Screenshot](image_url)

Figure 2.4: Screenshot of SourceForge.net (12)

Each project hosted on SourceForge.net may provide file releases to the community using File Release System. Files which have been released by a project will be available for users to download by locating a project which user interests select a file from the interested project and waiting for the file to be transferred.

More information about SourceForge.net is available on Internet. (11)
2.5. Related Works

I work on the project with another student, Henrike. It was a synchronous work. We mainly communicate by e-mail. We matched our projects UNIX name and we decided work on same Java projects.
3. Description of the Collected Data

This chapter describes the collected data while I am working on this project.

The following information is collected for each project: UNIX name, description of the projects, number of the developer, database environment, development status, intended audience, license, operating system, programming language, topic, translation, user interface, registered date, activity percentile, downloaded date and compiling status. In the following it shows the description of those collected data:

- **UNIX name**: UNIX name is used for project unique URL. It is the most important key for each project. It is a string value and users can type maximum 15 characters as an UNIX name. In SourceForge.net, when users type the UNIX name for new project, the system suggests to user UNIX name and it is checked by the SourceForge.net system.

- **Description of the project**: While users are creating a new project, description of the projects are shown as public description. It briefly summarizes the project. While users are browsing for any project, description of the project will be shown on the projects web page. It is a string value and it has to be between 20 and 255 characters.

- **Number of the developers**: It is an integer value. It shows how many developers work on the selected project.

- **Database environment**: It is a string value. When user creates new project, user selects database environment from the table which is suggested by SourceForge.net system. Database environment possibilities could be seen in Figure 3.1; database environment includes three levels, Database API, File Based DBMS and Network Based DBMS and also these levels include different levels.

![Figure 3.1: Database Environment](image)

- **Development Environment**: It is a string value. There are 7 possibilities for development status. The possibilities are shown in the following: 1-Planning, 2-Pre-Alpha, 3-Alpha, 4-Beta, 5-Production/Stable, 6-Mature, 7-Inactive.

- **Intended audience**: It is a string value. It is created by user. SourceForge.net suggests intended audience for users. Some intended audience possibilities could be seen in Figure 3.2 in different levels.
• **License:** It is a string value. It shows the license of the selected project. In order to host project on SourceForge.net, users must select one of the following open source software licenses; GNU General Public license (GPL), GNU Library or Lesser General Public License (LGPL), BSD License, Public Domain, MIT License, Apache License V2.0, Artistic License, Mozilla Public License 1.1 (MPL 1.1), Academic Free License (AFL).

• **Operating system:** It is a string value. When users of SourceForge.net want to create Project, they select operation system from the suggestion list. Some Operating systems could be seen in Figure 3.3.
• **Programming language:** It shows the programming language which developers use for selected project. There are so many possibilities but for my project Java is the main programming language.

• **Topic:** It is a string value. It is used for the description of category of Software projects. Some topic possibilities for software projects could be seen in Figure 3.4.

![Topic Diagram](image)

**Figure 3.4: Topic**

• **Translation:** It shows the languages for projects. The most used language is English for software projects.

• **User Interface:** It is string value. Some user interface possibilities for software projects could be seen in Figure 3.5.

![User Interface Diagram](image)

**Figure 3.5: User Interface**

• **Registered date:** It is date value. It shows the registration date for the selected project. It shows the date as “yy/mm/dd hh/mm”. The system takes this value automatically.

• **Activity percentile:** It is an integer value. It shows activity percentile of the project for last week. If users want to have a look for different time of activity percentile, they can see it from “View project activity statistics”.

• **Downloaded date:** It is date value, it shows the date which I downloaded selected project. I wrote down by hand.

• **Compiling status:** There are two possible values which are Yes/No. It shows whether the project is compiling. It is almost “Yes” selected.

More information about project details is available on Internet. (13)
4. Process Description

This chapter describes the process flow of the collection, preparation and analysis of my projects in different parts. I will also describe the process of this project.

Eclipse, VizzAnalyzer and SourceForge.net are main tools while studying on this project. As I discussed before in Background section, Eclipse and VizzAnalyzer have to be installed and SourceForge.net has to be online. In the Appendix A part you can find information about installation of Eclipse and VizzAnalyzer.

In my project, the task is to define a process for obtaining, storing and maintaining a large number of open software projects, to create a repository containing a large number of projects and to prepare them for analysis with the VizzAnalyzer and to extract software metrics. The data shall be provided for later analysis. The process needs to be automated as far as possible, repeatable, transparent, extendible and flexible. The efforts need to be evaluated. (2) To achieve this task at first we need to know what we get at the end of this project: workspace, 60-80 Java projects, snapshot for Java projects, database file from VizzAnalyzer and Entity-Relationship diagram of the database analysis result, changes report.

4.1. Downloading Software Projects

To begin with, I need to download 60-80 projects to analyze. SorceForge.net is helpful for this because it is the world’s largest open source software development web site.

There are couples of way to download software projects from SourceForge.net. The easy way of getting project is to use “Search” or “Browse Software” buttons. To write the words for interested project and click to the “Search” button will finish the process for finding project. It sorts the project by comparing the typed word and project name. It will return the project with descriptive name. And then user can select the project which is interested. Project web page will be shown like Figure 4.1; we can see the name of the project in bold font style like “Taste”, a download button with green color and some information about the project bottom of the download button.

![Figure 4.1: A software project web page](image)
By clicking the green button called “Download project name”, users see the file releases which could be seen in Figure 4.2. To select file you interested from the file releases will download the file.

The other way of downloading project is to use “Code” menu which is seen in Figure 4.1 and click “SVN Browse” or “CVS Browse” submenus. After clicking SVN or CVS it will present a window like Figure 4.3. We can see files of the project and download them by clicking. User can get projects from SourceForge.net by using those ways of downloading.

The next step is to collect all downloaded software projects in a repository. The repository is called Snapshot includes 60-80 Java projects. The software projects in this folder will be unmodified. With this folder, we can go out from the projects and produce them again for the case like making mistake or something. It is original reference source for the software projects for the reason that the software projects can change in the web and so it is not possible to regenerate them again by downloading from the web site. It might be a new version of the project.

4.2. Compiling and Analyzing Software Projects

It describes the next main step of this project that is compiling and analyzing software projects.

It starts with collecting all projects in another folder called workspace by coping them from the Snapshot folder. In order to analyze software projects with VizzAnalyzer, the projects should be compiled. When we get downloaded software projects from the web, probably they have some problems that prevent them to compile.

Eclipse shows the problems with description in “Problems” view. This view could be seen in Figure 4.4: we can find the code which has problem by clicking “Go To”.

<table>
<thead>
<tr>
<th>File</th>
<th>Rev</th>
<th>Age</th>
<th>Author</th>
<th>Last log entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>src/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lib/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>code/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHANGES</td>
<td>1.14</td>
<td>7 weeks</td>
<td>arowen</td>
<td>Updates for 1.7.2</td>
</tr>
<tr>
<td>LICENSE</td>
<td>1.1.1.1</td>
<td>14 months</td>
<td>arowen</td>
<td>Initial import</td>
</tr>
<tr>
<td>README</td>
<td>1.1.1.1</td>
<td>14 months</td>
<td>arowen</td>
<td>Initial import</td>
</tr>
<tr>
<td>build.properties</td>
<td>1.15</td>
<td>7 weeks</td>
<td>arowen</td>
<td>Updates for 1.7.2</td>
</tr>
<tr>
<td>build.xml</td>
<td>1.29</td>
<td>2 months</td>
<td>arowen</td>
<td>Added Recorder.isFiltered()</td>
</tr>
<tr>
<td>pom.xml</td>
<td>1.17</td>
<td>7 weeks</td>
<td>arowen</td>
<td>Updates for 1.7.2</td>
</tr>
</tbody>
</table>

Figure 4.3: CVS and SVN
Commonly, Java projects have missing library, syntax error and dirty workspace. As I discussed before Eclipse includes some features for Java projects like: Syntax highlighting, content/code assist, code formatting, import assistance, quick fix and integrated debugging features. To use those features is helpful to fix the project easily. Sometimes we need to search on Internet for missing libraries, classes’. After finding the missing elements; we can easily integrate it by using “Build Path”. I did not make changes in the code part while compiling the software projects. Sometimes it is not possible to be able to compile software projects because of the unknown problems. While I am compiling, I write down all changes as a report which will be described in coming part.

The next step after compiling all downloaded project is to analyze them. At first I define a database which works with VizzAnalyzer for storing data by using data source administrator. We can run VizzAnalyzer by clicking the VizzAnalyzer button on the perspective view; we can see it in Figure 4.5. When the cursor comes to on it, “Export CMM10 to DB” is shown.

![Figure 4.4: Problems View](image)

![Figure 4.5: VizzAnalyzer](image)

After clicking on VizzAnalyzer button, a window will appears for defining UNIX name of software projects for analyzing. Sometimes it is possible to one project includes two part in it. In this case, we can group selected projects and give same UNIX name. After typing UNIX name, we can move to next step by clicking “Next” button. The next step is to select the directory for graph configuration and export options. Finally, we have to fill all mandatory information and click to “Finish” button. Analyzing steps could be seen in Figure 4.6 and 4.7. You can see the projects name and select the project which you want to analyze in Figure 4.6. Figure 4.7 shows some mandatory project details which has to be filled for analyzing.
4.3. Documentation and Figure of the Process Flow

While doing on this project, I get some documents. This part summarizes these documents:

- **Project Detail:** It is an access and excel file. SourceForge.net provides user to get information about the project as you can see in Figure 4.8. The information for this document is taken from SourceForge.net. Some of these information is number of project developers, intended audience and license as you can see in Figure 4.7.
Changes Report: It is an access and excel file. It includes all changes while compiling projects. In this file, all changes are classified.

Database: It is access file. It is created by VizzAnalyzer at the end of the analysis.

Entity-Relationship diagram of diagram: It is a .jpeg file.

All documents could be found in separated compact disc. Figure 4.9 summarizes all process flow of this project. Firstly, we download the project from SourceForge.NET and create a repository and database with this projects and than create a new repository and make them compiled by using Eclipse. During the changes of the projects, create a changes report and than use VizzAnalyzer for analyzing all projects and create analyze result.
Figure 4.9: Process Flow
5. Analysis Result
This chapter describes the collected Java projects, analysis result and problems.

5.1. Collected Projects
This part provides an overview of the software projects. We will see what we collect for the projects.

100 Java projects are collected for this project. Databases, changes reports and graphics are collected for each Java project. We can see a screenshot of workspace folder in the following Figure 5.1. It shows the projects with UNIX name. This figure is taken from the workspace. As you can see, there are 100 Java projects name like “abbot”, “ajct”, “alster”.

As we discussed before, when we select the projects we focused on Java projects. In SourceForge.NET we filtered according to software for getting Java projects. Some of them had problem about compiling, we solved these problems. At the end of process we create some folders which includes lots of files related with this project, it could be seen in Figure 5.2:

- Analyze Drafts: It includes the result of the analyze for each project: .gml file, .log file and .txt file.
- Changes Report: It includes .pdf and excel sheets for describing the changes.
• **Database:** It includes access file which is given by VizzAnalyzer.

• **Projects information:** It includes .pdf excel and access file.

![Folder structure](image)

*Figure 5.2: Screenshot of the folders related with projects*

• **Snapshot:** It include all downloaded, unmodified 100 Java projects.
  More information about project is available on the given compact disc and Appendix A. In compact disc the description of all files are available in pdf files.

### 5.2. Analysis result and Problems

This part summarizes problems that Java projects have while compiling and analysis results.

At the end of the analysis, VizzAnalyzer gives gml file, log file and database file as draft. The files include software metrics values, projects information, and Java properties for each project. The drafts could be found in given compact disc.

Missing library, already defined type, syntax error, ambiguous type are some of the problems. In the following Figure 5.3 shows problem statistic for this project.
• **Missing Library**: The library consists of a lot of program. While compiling the project, if Eclipse can’t find the library, problems appear. Missing library is the most common problem that Java projects have. Users can download Java libraries from the web and build path as a solution.

• **Dirty Workspace**: Sometimes, Java projects can have dirty workspace. In this case to use “Project” and “Clean” menu will discard all build problems. The projects will be rebuilt from scratch by this process.

• **Already defined type**: Some of the project can have the same class or another type, it cause of the problem. The way of the solution is to delete one of the defined types or use another suggestion of Eclipse.

• **Syntax error, invalid expression**: It is quite easy to accidentally type something incorrectly while typing code. In this situation Eclipse will warn a syntax error. At the same time it can warn different kind of error, because it causes different problems like; “class” or “interface” expected, but the main problem is just syntax error.

• **Ambiguous type**: One of the compiling errors is ambiguous type. Somewhat unexpected errors can result when overloading a method that takes an argument. These errors involve ambiguity because it is possible to create an ambiguous call to an overloaded method. Some part of code has to work with another part in coordinate. If it doesn’t happen, it will show ambiguous type error.

• **Unmatched declared package**: In Java, you can group a bunch of classes into something called a package. Sometimes it is possible to happen that some package defined in the code and existing in the project are unmatched. In this case, the solution is to move or rename the package.
• **Unimplemented method:** A class that implements an interface must define the methods of the interface. Sometimes it is possible to unmatched with the method. Sometimes, inheritance cannot be correct defined and it can cause this kind of problems.
6. Conclusion and Future Work

This last chapter reflects on the result of the thesis. It shows how the problem described in the introduction has been solved and if all the goals have been met and the criteria fulfilled. It further discusses possibilities and ideas as to how the outcome of the thesis could be improved and directions of the future developments.

6.1. Conclusion

The problem addressed by this thesis is a defining task as described in the introduction section; define a process for obtaining, storing and maintaining a large number of open software projects, create a repository containing a large number of projects and to prepare them for analysis with the VizzAnalyzer and extract software metrics. The solution for solving this problem is based on defining a framework for process flow.

I described the goals and criteria for solving the problem in the introduction section. I can show the result of my work as Figure 4.9 and in the given compact disc.

The first goal was to get well understanding of the Eclipse and Java. This goal was met by the course called “Object oriented and Software Engineering” from Vaxjo University and also by reading books and searching on Internet. After getting this information, it helps me to compile projects for VizzAnalyzer as you can see in process description chapter.

The second goal was to get knowledge about SourceForge.net. This goal was met by using SourceForge.net and reading the instructions on web site. It is fulfilled for the reason that I downloaded 100 Java projects from this website to repositories for analyzing.

The last goal was to analyze Java projects by using VizzAnalyzer. This goal was met by analyzing 100 Java projects. All goals are met by defining a process flow for all projects and reporting some draft discussed in the related parts. The result of this project is given in a compact disc and report.

6.2. Future Works

In order to fulfill improved projects outcome, I should focus on the following things.

While working on this project, I used for downloading green button. Future search can be made for the other download ways. So we can see if there is a different.

Future search can be made for the Java projects and the other programming language. Actually in my projects some of projects programming language is not just Java, it is Java and other languages. I want to work with different projects like a project is written in Java and another programming language.

Another future work can be made for compiling projects before analyzing them. The project can find missing library and try to fix it by adding library or for syntax errors it can be checked by Eclipse. I can improve this with a new process will make all of these process flow automatically.
Bibliography


Appendix A
This chapter describes how to install Eclipse platform, project description and some statistic figures about the problems and projects.

1. Installation of Eclipse Platform and VizzAnalyzer
Eclipse provides user to facility installation. Users can get it from Eclipse web site. When it is taken on computer by downloading, and then by double-clicking on the previously downloaded file, you can extract and see files. In this file Eclipse application will be seen with a blue .exe file. Figure A1.1 shows Eclipse folder and blue file in a red rectangle. Eclipse doesn’t have long installation process like other programs.

After double-clicking to blue Eclipse file, it will ask you to specify the workspace to use. Workspace will be a store for projects. The window for this selection could be seen Figure A1.2. After selecting workspace place by using Browse, we can select the check box called “Use this as the default and do not ask again”. The installation is over by clicking OK button. Welcome page will be shown as we can see in Figure A1.3.
Eclipse has common user interface for working with tools. It provides a standard user navigation model for users. It provides programmers to program easily with its user interface. In Figure A1.4, it is seen basically how it is look like: some tools on the top and on the left side we can see projects which we work on it and on the bottom problem description could be seen.
VizzAnalyzer is integrated with Eclipse, when we install Eclipse we will see VizzAnalyzer button will be shown.

2. Statistic Figures
In this part, some figures are given that shows statistic values about the software projects and projects problem.

Number of projects with problem and without problem could be seen in Figure A2.1 for 100 Java projects which I downloaded. 32 projects include problem when I download them.

Figure A2.1: Number of Projects with problem and without problem
In Figure A2.2, we can see number of projects and problems for each year from 1999-2008 for 100 Java project that I downloaded. From 2007, 5 times it occurred in 21 projects.

Figure A2.2: Projects and Problems for each years

In Figure A2.3, we can see problem types and numbers. Missing library has occurred 28 times as we can see in Figure A2.3.

Figure A2.3: Problem types and numbers
In Figure A2.4, missing libraries could be seen. According to this figure, Junit is the most needed library for this project.

3. Collected Projects

In this part, some figures are given that shows statistic values about the software projects and projects problem. This part provides an overview of the software projects which we can see in the following Table A.1. It includes number of projects, UNIX name and public description. We can get some information on this table about the projects.
<table>
<thead>
<tr>
<th>P.N.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>abbot: Abbot is a simple framework for unit and functional testing of Java GUIs. Facilitates generating user actions and examining component state. Supports recording and playback on any Java application.</td>
</tr>
<tr>
<td>2</td>
<td>ajct: Simple and extensible java configuration framework/component/API able to read XML, LDAP, dB, JNDI, system props, ini files... data as configuration sources. Supports a mixture of all. Also allows to export (parts of) configuration data in other formats.</td>
</tr>
<tr>
<td>3</td>
<td>alster: Alster is a unit testing framework for XSLT files written in XSLT itself (but with a dependency on Nalax, an XSLT extension package for Apache Xalan-J).</td>
</tr>
<tr>
<td>4</td>
<td>ant-doxygen: Ant-Doxygen is an Ant task for doxygen, a document generating tool available at <a href="http://www.doxygen.org">www.doxygen.org</a>. Users can integrate doxygen related code with ant task file - build.xml.</td>
</tr>
<tr>
<td>5</td>
<td>antlr-testing: ANTLR Testing is a JUnit-extension for unit testing grammars produced by ANTLR.</td>
</tr>
<tr>
<td>6</td>
<td>aoi: Art of Illusion is a full featured 3D modelling, rendering, and animation studio. It is written entirely in Java, and can run on almost any operating system.</td>
</tr>
<tr>
<td>7</td>
<td>autofetch: Autofetch automates adding association prefetches to object queries. Prefetch is an optimization to reduce the number of round-trips to an object store. Autofetch is designed to integrate with different object persistence technologies such as Hibernate.</td>
</tr>
<tr>
<td>8</td>
<td>blogunity: Blogunity is a community blogging software written in Java.</td>
</tr>
<tr>
<td>9</td>
<td>buddi: Buddi is a simple budgeting program targeted for users with little or no financial background. It allows users to set up accounts and categories, record transactions, check spending habits, etc.</td>
</tr>
<tr>
<td>10</td>
<td>bugtracker: The Bugtracker is a complete bug/issue management system, built around MySQL utilizing Java Servlets for a web based front end. This will soon grow into a complete project development architecture encapsulating project management, issue tracking, QA, etc.</td>
</tr>
<tr>
<td>11</td>
<td>cdbrowser: Catalog and browse files on CD and HD, also searches on edonkey / emule network and saves the search. Indicates if an edonkey / ed2k file is on some CD and helps locating similar content.</td>
</tr>
<tr>
<td>12</td>
<td>classviewer: Get constructor, field, method information with free search for methods and option to view javadocs at method with your preferred browser. Configured by XML document.</td>
</tr>
<tr>
<td>13</td>
<td>cleanj: CleanJ is ABC-to-Java translator. It enables concurrent clean programs to run on JavaVM.</td>
</tr>
<tr>
<td>14</td>
<td>craftyftp: CraftyFTP is a FTP client written in Java. It provides a graphical interface closed to SmartFTP interface.</td>
</tr>
<tr>
<td>15</td>
<td>dbunit: DbUnit is a JUnit extension targeted for database-driven projects that, among other things, puts your database into a known state between test runs.</td>
</tr>
<tr>
<td>16</td>
<td>dcraw4j: DCRaw4J converts digital camera RAW image files into standard formats such as JPEG and 16-bit TIFF (linear or non-linear). DCRaw4J was inspired by David Coffin's dcraw; a C program that handles raw formats from over 200 cameras.</td>
</tr>
<tr>
<td>17</td>
<td>djdoc: DJDoc is a Java based web application for personalized access</td>
</tr>
<tr>
<td></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>---</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>18</td>
<td><strong>dlog4j</strong>: DLOG4J provide a full-featured, cross-platform, multi-database, wap-supported blog software, based on JSP/Servlet.</td>
</tr>
<tr>
<td>19</td>
<td><strong>eje</strong>: EJE is a simple Java editor, perfect to learn Java, without learning a complex development tool. EJE is multi-platform (written in Java), light-weight, user-friendly and have several useful basic features. A good help to start with Java!</td>
</tr>
<tr>
<td>20</td>
<td><strong>essence</strong>: Essence is a simple Java framework with modules for Quantitative Analysis and Distributed Data systems.</td>
</tr>
<tr>
<td>21</td>
<td><strong>exte</strong>: EXTE recognizes temporal expressions in a text. Temporal expressions are any references to the TIME e.g.: &quot;today&quot;, &quot;12th march 2006&quot;, &quot;as soon as he woke up&quot;. This analyzer will recognize three languages: Spanish, English and French.</td>
</tr>
<tr>
<td>22</td>
<td><strong>ezmorph</strong>: EZMorph is simple java library for transforming an Object to another Object.</td>
</tr>
<tr>
<td>23</td>
<td><strong>fitdecorator</strong>: Framework for writing decorator for FIT [Framework for Integrated Test][<a href="http://fit.c2.com/">http://fit.c2.com/</a>] fixtures. This project also provides a set of standard fit fixtures which can be used to decorate existing FIT table without modifying them.</td>
</tr>
<tr>
<td>24</td>
<td><strong>freecol</strong>: FreeCol is an open version of Colonization. It is a Civilization-like game in which the player has to conquer the new world.</td>
</tr>
<tr>
<td>25</td>
<td><strong>granite</strong>: Granite Data Services is a free, open source ( LGPL'd), alternative to Adobe® LiveCycle® (Flex™ 2) Data Services for J2EE application servers with EJB3/Spring/Seam/Guice/Pojo services support. See details on <a href="http://www.graniteds.org">http://www.graniteds.org</a>.</td>
</tr>
<tr>
<td>26</td>
<td><strong>gridiron</strong>: A simple web application framework written in java using the Servlet 2.2 API. Gridiron helps to rapidly develop MVC web applications by giving a direct mapping between actions and views and supplying a Navigation class to aid in the development of views.</td>
</tr>
<tr>
<td>27</td>
<td><strong>hsqldb</strong>: HSQLDB is a relational database engine written in Java, with a JDBC driver, supporting a large subset of ANSI-92 SQL. A small, fast engine with both in memory and disk based tables. This product is the continuation of HypersonicSQL. Active since 2001.</td>
</tr>
<tr>
<td>28</td>
<td><strong>hisim</strong>: Create and simulate Hybrid Petri Nets.</td>
</tr>
<tr>
<td>29</td>
<td><strong>htformvalidator</strong>: Don't download this. It is very incomplete</td>
</tr>
<tr>
<td>30</td>
<td><strong>httpunit</strong>: a Java library for the automatic stimulation and testing of web applications</td>
</tr>
<tr>
<td>31</td>
<td><strong>invicta</strong>: Invicta is a build management tool. Using simple project definition files, it generates powerful build scripts (such as ANT) while hiding their complexity. Invicta is a modular framework that allows developing additional components and output types.</td>
</tr>
<tr>
<td>32</td>
<td><strong>j2ssh</strong>: The J2SSH libraries provide a platform independent, extensible implementation of the SSH2 protocol.</td>
</tr>
<tr>
<td>33</td>
<td><strong>jabble</strong>: A framework to solve partial differential field equations in Java.</td>
</tr>
<tr>
<td>34</td>
<td><strong>jailer</strong>: Model-based data export, schema browsing and rendering. Exports consistent, referentially intact row-sets from relational databases. Removes obsolete data without violating integrity. DBMS agnostic (JDBC). Generates topologically sorted SQL-DML.</td>
</tr>
<tr>
<td>35</td>
<td><strong>jaimlib:</strong> JAIM implements the AOL IM TOC protocol in an easy to use Java library. The primary goal of JAIM was to simplify writing AOL bots in Java but it could also be used to create Java based AOL clients.</td>
</tr>
<tr>
<td>36</td>
<td><strong>jalita:</strong> Java Light Terminal Adapter - a simple server framework to develop java applications for terminals (i.e. vt100) in a swing based manner, specialised for mobile devices typical used in business environments (i.e. barcode scanner) connected via tcp/ip.</td>
</tr>
<tr>
<td>37</td>
<td><strong>jarjar:</strong> Reduce Java dependency headaches by repackaging third-party jars.</td>
</tr>
<tr>
<td>38</td>
<td><strong>jasml:</strong> JASML is a java byte code compiler, providing yet another approach to view, write and edit java classes, even without the existence of a java source file - using the java macro instructions, those described in The Java Language Specification.</td>
</tr>
<tr>
<td>39</td>
<td><strong>jasperreports:</strong> JasperReports, the world's most popular open source business intelligence and reporting engine and JasperServer, the most popular open source interactive report server built-on JasperReports.</td>
</tr>
<tr>
<td>40</td>
<td><strong>javacrpg:</strong> Developing classic RPG framework with Java &amp; OpenGL. An attempt to create a detailed, old-fashioned pen'n'paper like &amp; turn based RPG with a decent story/AI and a huge place to play. Also includes a reference RPG. Blog: <a href="http://jcrpg.blogspot.com">http://jcrpg.blogspot.com</a></td>
</tr>
<tr>
<td>41</td>
<td><strong>javadocpicker:</strong> DatePicker and DateEdit are Swing components that let you easily add calendar functionality to any Java application. Both components extend JPanel and can be incorporated into dialogs, frames and other Swing components that support JPanel components.</td>
</tr>
<tr>
<td>42</td>
<td><strong>javamms:</strong> Download MMS served files to your PC. Now, no matter what your connection speed you can watch good quality streamed media files without continual dropouts.</td>
</tr>
<tr>
<td>43</td>
<td><strong>javaquantity:</strong> This Java library (currently 217 classes) is able to handle quantities in terms of basic arithmetics, conversion to similar units, serialization. As framework it allows to create new dimensions and units by inheritance.</td>
</tr>
<tr>
<td>44</td>
<td><strong>javavis:</strong> A Computer Vision Library in Java</td>
</tr>
<tr>
<td>45</td>
<td><strong>jconfig:</strong> The goal of this project is to build a configuration manager in Java that works with xml files, an URL or a LDAP repository to read the configuration properties. MBean / JBoss support is now available.</td>
</tr>
<tr>
<td>46</td>
<td><strong>Jcryptfx:</strong> Java Crypt Framework (JCryptFX) is a java-based education project for de-/encrypting streams. It has a implementation of DES and TripleDES in it but the framework user can write own cryptography algorithms and stream types for it - includes a console app</td>
</tr>
<tr>
<td>47</td>
<td><strong>Jena:</strong> Jena is Java toolkit for developing semantic web applications based on W3C recommendations for RDF and OWL. It provides an RDF API; ARP, an RDF parser; SPARQL, the W3C RDF query language; an OWL API; and rule-based inference for RDFS and OWL.</td>
</tr>
<tr>
<td>48</td>
<td><strong>Jfish:</strong> JFish is a simple Java implementation of the popular Twofish cryptography algorithm. It allows any application to easily implement powerful encryption, using a single small library and minimal code changes.</td>
</tr>
<tr>
<td>49</td>
<td><strong>jmario:</strong> JMario is a side-scroll game, built in Java, based on Nintendo’s Super Mario World. It’s been developed by David Buzatto as a part of him’s course conclusion work entitled &quot;Utilization of Java Platform for 2D Game Development&quot;.</td>
</tr>
<tr>
<td>50</td>
<td><strong>jmemorize</strong>: jMemorize is written in Java and uses Leitner flashcards to make memorizing facts not only more efficient but also more fun. jMemorize manages your learn progress and features categories, Unicode flashcard texts, statistics and an intuitive interface.</td>
</tr>
<tr>
<td>51</td>
<td><strong>jncl-api</strong>: The Java Nessus Client API is a Nessus Transfer Protocol 1.2 Implementation. The JNC-API includes: NTP 1.2 Specification (.en, .de), Java API Documentation (.en), JNC-API Binary / Source, Handbook (.de), GUI Nessus Client (Source Only, Prototype)</td>
</tr>
<tr>
<td>52</td>
<td><strong>jnlp</strong>: A high-quality JNLP client that can also be used as a component to provide applications with Java Network Launching Protocol features and support. Netx downloads code over the network, caches it, and runs it in a secure sandbox environment.</td>
</tr>
<tr>
<td>53</td>
<td><strong>jogre</strong>: Java based client / server online games engine &amp; API for easy creation/running of real-time multiplayer games. Games include Battleships, Camelot, Checkers, Chess, Connect 4, Dots, Go, Octagons, Texas Hold'em Poker, Reversi, Spades, Tetris &amp; Tic-tac-toe.</td>
</tr>
<tr>
<td>54</td>
<td><strong>joustim</strong>: joscar is an easy-to-use, robust library for connecting to AOL Instant Messenger from Java using the Oscar protocol. Supports IM images, file transfer, and more.</td>
</tr>
<tr>
<td>55</td>
<td><strong>jpdfunit</strong>: JPdfUnit integrates PDFBox as an PDF API with the JUnit framework for the test of pdf documents so JPdfUnit is a high level api. Simple ready-to-use assertions help to compare the expected data to the concrete data of the pdf document.</td>
</tr>
<tr>
<td>56</td>
<td><strong>jsap</strong>: JSAP provides simple but powerful command-line parsing functionality for Java applications. Features include typed arguments, automatic parsing of multi-value arguments, user-defined argument classes, and automatic generation of usage and help text.</td>
</tr>
<tr>
<td>57</td>
<td><strong>jskat</strong>: JSkat is an implementation of the German card game Skat in Java. You can play according the rules of the German Skat Association and with some add on rules that are played in the pubs (e.g. Ramsch). All data are stored in XML.</td>
</tr>
<tr>
<td>58</td>
<td><strong>jsqsh</strong>: JSqsh is a Java implementation of sqsh, a console based database query tool featuring much of the functionality provided by a good shell, such as variables, redirection, pipes, history, command line editing, and dynamic configuration.</td>
</tr>
<tr>
<td>59</td>
<td><strong>jstester</strong>: JsTester allows validation of javaScript code inside java. It provides a group of assert methods like JUnit's Assert, it also supports the validations in <a href="http://javascript.crockford.com">http://javascript.crockford.com</a>, and the ability to use your own validations.</td>
</tr>
<tr>
<td>60</td>
<td><strong>jtpcc</strong>: Open Source Java implementation of the TPC-C benchmark.</td>
</tr>
<tr>
<td>61</td>
<td><strong>juiceforge</strong>: JUICE is a cross-platform utility for editing Aleph One map files. It can’t replace a dedicated editor like Forge, but it is capable of editing several aspects of map files, most of which are geared toward fixing and tweaking network maps.</td>
</tr>
<tr>
<td>62</td>
<td><strong>jvs-vfs</strong>: Java Virtual System (JVS) is built around a virtual file system (VFS) - a common platform of Java, by Java, for Java. JVS extends JVM and makes it from an interpreter running half-compiled code a first-class server.</td>
</tr>
<tr>
<td>63</td>
<td><strong>Jwebchart</strong>: Simple, easy to use Charting servlet developed using JFreeChart. Simply install in your favourite application server and pass data using GET and POST. Sit back and see your chart rendered in beautiful technicolor! No hassle charting using JFreeChart.</td>
</tr>
<tr>
<td></td>
<td><strong>jxarcade:</strong> JXArcade is a 2D Game Editor for JAVA. It allows you to produce Arcade java games using a graphic Interface to define sprites, paths, scene, etc.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td><strong>jzonic:</strong> jZonic is a Java framework for web-application supporting an XML workflow, XML sitemap management, connection pooling, uses jConfig, build upon the MVC architecture and include several demo applications</td>
</tr>
<tr>
<td></td>
<td><strong>kolmafia:</strong> KoLmafia is a cross-platform desktop tool which interfaces with the online adventure game, Kingdom of Loathing. KoLmafia is written in Java (J2SE 1.4 compliant), with binary releases in JAR format.</td>
</tr>
<tr>
<td></td>
<td><strong>latexdraw:</strong> LaTeXDraw is a graphical opensource PSTricks code generator or a PSTricks editor for LaTeX distributed under the GNU GPL. LaTeXDraw is developed in java, so it's independent of the OS.</td>
</tr>
<tr>
<td></td>
<td><strong>liquibase:</strong> A tool to manage database changes and refactorings. All changes to a database are stored in XML files that are stored in version control with other source code. A graphical IDE is also available.</td>
</tr>
<tr>
<td></td>
<td><strong>macad:</strong> A simple application to administer Magic: The Gathering trading cards, esp. for fast manual input of german/english cards. At the moment this application is limited to jdk1.5</td>
</tr>
<tr>
<td></td>
<td><strong>macpackage:</strong> Mac-Package is an Ant task and Maven goal for packaging native OS X applications.</td>
</tr>
<tr>
<td></td>
<td><strong>mp3dings:</strong> Mp3dings is a Java mp3 tagger and organizer with a clean spreadsheet-style user interface that permits processing lots of files in parallel. In addition, mp3dings has powerful automation capabilities. ID3v1 tags and most of ID3v2 are supported.</td>
</tr>
<tr>
<td></td>
<td><strong>mypod:</strong> The aim of the myPod project is to provide an platform indepedened application to have fun with an Apple iPod. myPod helps you to manage your music and to synchronize clips and playlists to your iPod.</td>
</tr>
<tr>
<td></td>
<td><strong>ocwfinder:</strong> OCW Finder is a search tool to find different OpenCourseWare material.</td>
</tr>
<tr>
<td></td>
<td><strong>openicq:</strong> OpenICQ is a simple ICQ-Clone written in JAVA programming language. OpenICQ comes up with support for multiple languages and user profiles.</td>
</tr>
<tr>
<td></td>
<td><strong>opensed:</strong> OpenSearch editor is the editor of Firefox/IE7 search's plugins. For more information, please, visit <a href="http://opensed.sourceforge.net">http://opensed.sourceforge.net</a></td>
</tr>
<tr>
<td></td>
<td><strong>paperclips:</strong> PaperClips - Simplified Java printing support for SWT. PaperClips hides the complexity of laying out and rendering documents on the printer, helping you focus on <em>what</em> to print instead of <em>how</em> to print it.</td>
</tr>
<tr>
<td></td>
<td><strong>pdfbox:</strong> PDFBox is a Java PDF Library. This project will allow access to all of the components in a PDF document. More PDF manipulation features will be added as the project matures. This ships with a utility to take a PDF document and output a text file.</td>
</tr>
<tr>
<td></td>
<td><strong>picweb:</strong> Photo Database / Album Web in three flavours - J2EE web app, PHP web app and offline generation of static html (Java)</td>
</tr>
<tr>
<td></td>
<td><strong>plexian:</strong> Storage Java library for autocomplete fields in Ajax applications.</td>
</tr>
<tr>
<td></td>
<td><strong>r-ant:</strong> r-ant is a small tool which allows you to run Ant build tasks</td>
</tr>
</tbody>
</table>
consecutively, without having to restart the Java Virtual Machine each time.

81 **regexsearch:** RegexSearch is a Java application that searches multiple files for specified text or a regular expression. Files are displayed in a scrollable window with the matched text highlighted. Occurrences of the matched text can be replaced singly or globally.

82 **robocode:** Robocode is a Java programming game, where the goal is to develop a robot battle tank to battle against other tanks. The robot battles are running in real-time and on-screen. The motto of Robocode is: Build the best, destroy the rest!

83 **runcc:** RunCC is a new kind of parser generator that generates parsers and lexers at runtime. Source generation is only optional. It features the absence of any cryptography. Although intended for small languages, it comes with Java and XML example parsers.

84 **sablecc:** SableCC is a parser generator which generates object-oriented frameworks for building compilers, interpreters and other text parsers. SableCC keeps a clean separation between machine and user code which leads to a shorter development cycle.

85 **sasax:** The name, "SASAX", is from "Simple API for SAX(Simple API for XML)". SASAX is the framework to parse XML document easily under SAX framework.

86 **simplepdf:** As the name of the application suggests, its very simple. User just have to provide the list of images and text files, whose contents they want to have in the PDF in a configuration file. The application reads the configuration file and generates the PDF.

87 **simpleweb:** Simple is a truly embeddable open source HTTP engine. It provides a high performance HTTP engine with a component based API for extendability and integrates seamlessly with XSLT and the Velocity template engine facilitating an MVC architecture.

88 **solinger:** Solinger contains Java ports of useful C libraries. This includes a small/fast pure Java dbm (sdbm) and a Java password checker (CrackLib).

89 **storytestiq:** StoryTestIQ (STIQ) is a test framework used to create Automated Acceptance Tests. STIQ is a mashup of Selenium and FitNesse: its "wiki-ized" Selenium with widgets and features that make it easier to write and organize Selenium tests.

90 **sudoku:** A Java project to solve the Sudoku puzzles (see http://www.sudoku.com) that appear daily in most newspapers. The solver applet, which explains exactly how it arrives at its solution, is hosted at http://act365.com/sudoku

91 **suprsukr:** suprsukr is a data aggregator which can read data from rss, atom, oai and microformats.

92 **sywico:** SyWiCo is a tool for managing concurrent modifications of shared files between unconnected computers. It can be used as a synchronization tool relying on email.

93 **taste:** The collaborative filtering engine for Java. The engine takes users' preferences for items ("tastes") and returns estimated preferences for other items. Engine is scalable (millions of preferences) and flexible (can selectively replace implementations).

94 **the-egg:** EGG is a development tool for fast and easy realization of XML to plain-text tranformations. It's similar to XSLT, Velocity or
JSP but emphasises source-code generation in a model-driven, generative development-process. For Windows or Unix with Java.

| 95 | trmi: Transparent RMI is an extension to Java's RMI mechanism that greatly simplifies the creation of distributed applications. It allows any interface to be used remotely, and provides centralised recovery from RemoteExceptions. |
| 96 | weirdx: WeirdX is a pure Java X Window System server. |
| 97 | wicket: Apache Wicket is a component based Java web application framework. It has found a new home at Apache: http://wicket.apache.org Please visit us there to discover new releases, find support and learn more about Apache Wicket. |
| 98 | workingcode: WorkingCode gives you what the name says working pieces of codes in Java. We are talking about code snippets and full utility apps for Java and J2me. It's use is intended as a library. |
| 99 | xdc: Solinger contains Java ports of useful C libraries. This includes a small/fast pure Java dbm (sdbm) and a Java password checker (CrackLib). |
| 100 | Zirc: An irc client written in pure java. It supports multi-server connections, dcc file sharing, mirc-compatible text coloration, smilies, a simple, intuitive interface... It has everything you may need to make your irc experience easy and enjoyable |

*Table A.1: Collected Java Projects (13)*