

Cleveland State University
EngagedScholarship@CSU



ETD Archive

2011

The Relational Database: a New Static Analysis Tool?

Adam M. Dutko
Cleveland State University

Follow this and additional works at: <https://engagedscholarship.csuohio.edu/etdarchive>

 Part of the [Electrical and Computer Engineering Commons](#)

How does access to this work benefit you? Let us know!

Recommended Citation

Dutko, Adam M., "The Relational Database: a New Static Analysis Tool?" (2011). *ETD Archive*. 733.
<https://engagedscholarship.csuohio.edu/etdarchive/733>

This Thesis is brought to you for free and open access by EngagedScholarship@CSU. It has been accepted for inclusion in ETD Archive by an authorized administrator of EngagedScholarship@CSU. For more information, please contact library.es@csuohio.edu.

**THE RELATIONAL DATABASE: A NEW STATIC
ANALYSIS TOOL?**

ADAM M DUTKO

Bachelor of Chemistry

Davidson College

August, 2003

submitted in partial fulfillment of the requirements for the degree

MASTERS OF SCIENCE IN SOFTWARE ENGINEERING

at

CLEVELAND STATE UNIVERSITY

July, 2011

©Copyright by Adam M Dutko 2011

This thesis has been approved for the
Department of **ELECTRICAL AND COMPUTER ENGINEERING**
and the College of Graduate Studies by

Thesis Committee Chairperson, Dr. Nigamanth Sridhar

Department/Date

Dr. Yongjian Fu

Department/Date

Dr. Wenbing Zhao

Department/Date

Redbird and Bean

THE RELATIONAL DATABASE: A NEW STATIC ANALYSIS TOOL?

ADAM M DUTKO

ABSTRACT

Code comprehension is pivotal to reducing errors in software. Reading source code improves code comprehension and enables effective fixes but as a code base grows meta-data become increasingly important. Static Analysis techniques provide an avenue for software developers to learn more about their code through meta-data while also helping them safely detect potential errors in their source. Unfortunately, many Static Analysis tools have a steep learning curve and are limited in scope. This thesis seeks to make Static Analysis accessible and extensible by asking what ubiquitous tools like SQL and relational databases can offer and what they cannot. We begin to answer these questions by exploring the source code of three C++ projects (libodbc++, log4cxx, C++ Sockets Library) using a new Static Analysis tool called Trike. Initial results indicate Trike is a promising and accessible tool for analyzing the structure of a code base. With further improvements, Trike should equal more established Static Analysis tools in scope and surpass them in usability.

TABLE OF CONTENTS

	Page
ABSTRACT	v
LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER	
I. Introduction	1
1.1 The Problem	2
1.2 The Thesis	3
1.3 The Solution Approach	3
1.4 Contributions	4
1.5 Organization of the Thesis	5
II. Related Work	6
2.1 Static Analysis	6
2.2 Relational Databases	10
2.3 Extensible Markup Language	10
III. Trike Architecture	12
3.1 The GCC-XML Extension	13
3.2 The XML Schema	16
3.3 The Database Schema	16
3.3.1 Mapping XML to SQL DDL Statements	20
IV. Static Analysis with Trike	21
4.1 C++ Projects Analyzed	22
4.2 Existing Trike Queries	24
4.2.1 Structural Queries	25
4.2.2 Composite Queries	35
4.2.3 Advanced Queries	41
4.3 Further Query Discussion	42

V.	Discussion	44
5.1	Potential Limitations	45
5.2	Areas for Improvement	45
	BIBLIOGRAPHY	47
	APPENDIX	54
A-1	Structural Query Output	55
A-1.1	The Number of Files	55
A-1.2	The Number of Header Files	56
A-1.3	The Number of Source Files	56
A-1.4	Total Source Lines of Code (SLOC)	57
A-1.5	The SLOC per File	58
A-1.6	Total Header SLOC	70
A-1.7	Total Source SLOC	70
A-1.8	The Bytes per File	71
A-1.9	The Number of Functions	83
A-1.10	The Number of Functions per Generated XML File	84
A-1.11	Total Class Count	90
A-1.12	Total Variable Count	91
A-1.13	Total Private Variable Count	92
A-1.14	Total Protected Variable Count	93
A-1.15	Total Public Variable Count	93
A-1.16	List of Included Library Files	94
A-1.17	Number of Included Library Files	105
A-1.18	File Weight	106
A-2	Composite Query Output	118
A-2.1	Average SLOC for all Files	118
A-2.2	Average SLOC for Header Files	119
A-2.3	Average SLOC for Source Files	120
A-2.4	Proportion of Private, Protected and Public Variables	120
A-2.5	Percentage Private Variables	121

A-2.6	Percentage Protected Variables	122
A-2.7	Percentage Public Variables	123
A-2.8	Set of Queries to Calculate Percentage of Files Referenced	124
A-2.9	Percent Code Reused	135
A-3	Advanced Query Output	136
A-3.1	Search Source for a String	136
A-3.2	Search XML for a String	139
A-3.3	Find Code Surrounding a String in Source	140
A-3.4	Find Code Surrounding a String in XML	141

LIST OF TABLES

Table		Page
I	Structural Query Output	35
II	Composite Query Output	40

LIST OF FIGURES

Figure		Page
1	Example Trike Database Relationships	18
2	Table Listing of Trike Database Schema	19

CHAPTER I

Introduction

Software was once considered a luxury because of its high cost and the relative scarcity of systems to run it on [4, 29, 46]. Some of the earliest adopters were banks and researchers [4, 29]. As these groups illustrated the utility of leveraging software to perform calculations and the cost of computers decreased, other groups began to push for the development of new software packages [4, 29]. This demand helped create a vibrant industry that today employs and services a large percentage of the world's population [4, 46].

When society relied less on software based systems, downtime was an inconvenience. Now that society depends on software based systems, errors are much more than a simple inconvenience [4, 20]. Our dependency on software makes writing correct and error free software essential to commerce and critical to humanity. So how is it that we find it so hard to eliminate errors in software [4, 20]? A consensus on why this is so difficult remains elusive. Ensuring software is free from errors has become the quintessential goal of the field of *Software Engineering* [46]. Learning how to reduce errors in software is a facet of this pursuit.

My thesis seeks to make existing error elimination techniques easier to use by leveraging a technology familiar to most developers: the relational database. I explore this new approach to the field of *Static Analysis* using the first SQL based

Static Analysis tool called Trike. The methods introduced by Trike, through the exploration of real C++ source code, should make it easier for software developers to extract meaningful meta-data from their C++ projects to help them understand how projects change over time. I believe this introduction serves as a good starting point for research into SQL based Static Analysis and defend this belief by discussing why the extracted meta-data should motivate developers to explore this new direction in the field of *Static Analysis*.

1.1 The Problem

Most modern software development processes provide developers with the opportunity to reflect upon errors, develop and implement patches and implement process improvements and tools to help prevent the same errors in future projects and steps in the development life-cycle [20, 46]. This approach to professional software development seems likely to persist until all software can be proven correct, in a trivial manner. Relying on software developers to discipline themselves to incorporate processes and tools into their work flow remains the only practical route to reducing errors in software [20, 29, 46]. Yet one of the highest barriers to adopting tools and processes is the level of effort required for developers to learn and integrate them into their work flow [3, 4]. If a tool or process is difficult to use it will most likely be avoided. If a tool is relatively easy to use it should have a naturally lower barrier to adoption and experience an increased level of adoption.

Each Static Analysis tool is unique and the constraints affecting the approaches tool developers take to build useful tools often elicits unique solutions. Some approaches are as unique as writing tool specific languages used to extract data and help identify errors while others are as simple as limiting the tool in scope and extensibility [5, 7, 8, 11, 13–15, 19, 26–28, 30, 32, 34–40, 42, 43, 45, 57–61]. When tool developers choose to rely upon tool specific languages they make their tool less appealing to busy developers and when they reduce the scope of their tool too much, the effort of using their tool eclipses any derived benefits. Without approachable and useful tools many

software developers will avoid the pain associated with integrating a Static Analysis tool into their work flow. This is something that Trike seeks to change for C++ developers.

1.2 The Thesis

The central question I seek to answer in my thesis is if a relational database can be used for Static Analysis. I arrived at this question because I wanted to make *Static Analysis* approachable to software developers and easy to integrate into their work flow. In my mind, the most effective way to accomplish these goals is to identify a seemingly unrelated tool with similar traits, such as a relational database, and then use it to build an approachable and powerful Static Analysis tool.

1.3 The Solution Approach

Before I started development on Trike, I explored existing Static Analysis tools and discovered that many of them were either expensive, complex, incomplete or a mixture of the three. Trike does not require a costly license, something that is easily accomplished through code reuse and extension using code from liberally licensed projects like GCC-XML and by making the interface for extending Trike as easy as writing a new SQL query. Trike is not complex, something that is accomplished by making the analysis language the same language in use by most developers to consume data from a database, SQL. Trike is capable, a tool that is currently able to extract useful meta-data about C++ projects.

I believe my solution approach addresses the three important issues affecting the adoption and use of Static Analysis tools today: cost, complexity and flexibility. Trike does not eliminate these issues but serves as a proof of concept showing the potential for using the relational database model to perform Static Analysis on C++ code. Some work remains to grow Trike to encompass other areas of Static Analysis. In the future, I might have to replace GCC-XML with another tool like CIL or make

modifications to GCC-XML to accommodate such enhancements but for now GCC-XML provides a useful starting point.

1.4 Contributions

My thesis asks if a relational database can be used to perform Static Analysis. I begin to answer this question by building a prototype of the first Static Analysis tool to leverage the relational database, called Trike. I then use Trike to show the expressiveness and utility of leveraging a relational database to gather structural meta-data on three different C++ programs (libodbc++, log4cxx, the C++ Sockets Library). In order to implement Trike, I had to write a GCC-XML wrapper capable of generating whole C++ program XML representations from individual source files and a unique database schema based on the Extensible Markup Language (XML) files generated by GCC-XML and constrained by an experimental Document Type Definition (DTD) for C++.

Overall, the implementation of Trike resulted in

- a wrapper program used to leverage existing code and technologies,
- a unified configuration file with settings for three example projects,
- an experimental C++ language database schema,
- a utility for generating and importing XML representations of C++ source code,
- queries used to illustrate the usefulness of Trike for three different C++ programs,
- and the implementation of the first *Static Analysis* tool to leverage a relational database.

1.5 Organization of the Thesis

My thesis has five major sections. The first section provides an “Introduction” and has five subsections that I use to state why reducing errors in software is important, ask if it is possible to use a relational database to statically analyze software, discuss my approach to answering that question, and outline my contributions to the field of *Static Analysis*. I use the second section to discuss “Related Work” in the field of Static Analysis as well as discuss relevant work related to relational databases and extensible markup language (XML). I use the third section to discuss the “Trike Architecture,” which includes discussion about the relevant sub-components, GCC-XML, the XML schema, the database schema and my approach to mapping XML to tables, using the SQL data definition language. I then use the next section, Section four, to discuss the bulk of my work. In this section I discuss Static Analysis from the context of Trike, identify three relevant C++ projects and discuss the results of the structural, composite and advanced queries I ran on their respective code bases. I conclude the fourth section with a discussion about why developers should find the results from these queries intriguing. Finally, I use the fifth section to discuss the potential limitations of Trike and areas where Trike could improve.

CHAPTER II

Related Work

It is surprising not to discover other researchers utilizing a database to perform static analysis on a code base. Other researchers have explored XML then used XQuery to check constraints while others have used XML as a means of generating an AST and analyzed the code base with programmable constraints [8, 47, 61]. To my knowledge, the use of a database and SQL to perform static analysis is unexplored until now. Trike is a new Static Analysis framework currently capable of extracting useful project meta-data. Trike is not at the level of other existing tools like the Clang Static Analyzer, cppcheck, cpplint and the Oink stack but with more work it should be and should have the added benefit of being accessible to beginning developers as well as being faster at regenerating the model on which query results are based [10, 25, 33, 56].

2.1 Static Analysis

Static Analysis is a conflated term and serves as an umbrella for the varied approaches to analyzing source code before it is executed [16]. Approaches to static analysis vary because not all languages are the same [16]. Some languages are strongly typed while others are weakly typed. Some languages support automatic memory management while others require manual memory management. Some language compilers gen-

erate byte code and operate atop a virtual machine while others generate machine code and operate directly atop the hardware. These disparities influence the possible errors that can exist in a particular base of code which when coupled with the syntactical differences between languages makes writing a single static analysis tool for all languages and errors, difficult, if not impossible.

Static Analysis techniques help developers gain a better understanding of their program by revealing important structural meta-data and detecting potential errors before they become larger problems. Using Static Analysis techniques, developers can discover errors that do not depend on values of variables and objects such as semantic errors, type errors, memory errors, logic errors, interface and include errors and various security errors associated with each [16]. Developers can also learn important structural data about their programs such as line count, character count, file count, class composition, and function count. Such meta-data not only enhance code comprehension but can also prove useful during code refactoring when generating code statistics becomes important. Some useful statistics that can enhance the process of code refactoring include the total lines of code for a project, the average lines of code per file, the number of functions per file, the number of classes per file, functions per line of code (or function density), classes per line of code (or class density), and the total number of system and library files used [16].

Static Analysis tools typically perform these operations using program models [16]. Such models vary with what data the tool has available, what errors and meta-data a tool is built to discover and the source language of the program(s) to be analyzed [16]. In order to construct the model, a Static Analysis tool must typically have access to the source code, the Abstract Syntax Tree (AST), the byte code or the compiled binary [16]. Although most tools rely on a model to analyze a program some tools do not require a model.

If the tool is built to discover errors, it might use language constructs (eg. variables, functions) or symbols to build a model such as a well connected graph of execution (with or without contextually sensitive data), then use the relevant aspects of the graph and tool specific logic to identify potential errors [16]. If the

tool is built to discover meta-data like structural information it might use the same language constructs but bypass model generation and simply calculate each relevant data point using heuristics or logic. If the tool is built to extract meta-data and identify potential errors it might take a divide and conquer approach, generating a model when it needs it and using heuristics and logic when it does not.

Trike is written to statically analyze C++ programs and requires the source code of the target program. The source code is required because Trike uses it as the basis for generating an XML model for each file which provide the basis for static program analysis in Trike. After a model is generated, it is parsed and loaded into the database element by element. The developer then writes SQL queries to perform static program analysis on their code.

A myriad of Static Analysis tools exist in the literature and industry. It is remarkable that none of the existing static analysis tools utilize a database and SQL to analyze source code. Of the tools that exist I chose to compare the Clang Static Analyzer, cppcheck, cpp lint and the Oink stack to Trike.

The Clang Static Analyzer is different from Trike for a few reasons. One of the most important is that it is targeted to discovering bugs in C and Objective C programs. The project is implemented in C++ but C++ is not currently a language targeted for analysis despite there being support for C++ in its tool-chain. Support for it should be added eventually and might already be ready to commit but no such support is listed on the homepage. In addition to this difference, the Clang Static Analyzer can either be run as a stand-alone tool or from within the Apple Computer Integrated Development Environment called Xcode [1, 33]. Trike is currently only a stand-alone tool. Another important difference is that the Clang Static Analyzer is restricted to performing a particular set of static analysis routines implemented in the program logic. In order to extend Clang a patch must be submitted [33]. The Clang Static Analyzer is not completely different from Trike. The Clang Static Analyzer also uses a compiler infrastructure and requires source code. Instead of using the GNU Compiler Collection (GCC), like Trike, it uses the Low Level Virtual Machine (LLVM) infrastructure to analyze source code. It is also open source, just like Trike.

cppcheck is another useful static analysis tool. cppcheck focuses on discovering a variety of well-known errors as well as adherence to some industry accepted coding guidelines in C and C++ code. Some of the errors and guidelines cppcheck looks for include checking for the use of auto variables, bounds checking, class attributes, exception safety, memory leaks, null pointers, obsolete functions, standard template library usages, uninitialized variables, unused functions and using postfix operators. While these are all useful errors and guidelines to check for the approach used by cppcheck is different from Trike. cpplint uses an executable with a limited set of predefined rules managed by the cppcheck developers. Like the Clang Static Analyzer, cppcheck can be integrated into an IDE such as Microsoft Visual Studio or the Eclipse Foundation called Eclipse [10,48]. There are also cppcheck plugins for several Continuous Integration tools like Hudson and Jenkins. cppcheck can be extended by modifying the code which just like Trike, is also open source.

cpplint, the tool released by Google, is a very light static analysis that mainly focuses on identifying code that does not adhere to the Google C++ style guide [25]. cpplint focuses on syntax errors and best practices according to Google. It performs these duties using various heuristics on source code and is open source like Trike. It is also written in Python, just like Trike.

Of all the tool surveyed, the most advanced is the Oink stack [56]. The Oink stack is “a collaboration of C++ static analysis tools” and utilizes the CQual solver as the basis for its CQual++ polymorphic whole-program data-flow analysis engine. Like Trike, Oink uses GCC as part of its tool stack. Oink is able to check for syntax errors and for the use of best practices defined by the developers. Oink is also extensible and composable enabling developers to add their own rules for analysis and add it to their own projects. Most importantly, Oink is able “to compute expression-level and type-level data-flow and statement-level intra-procedural control-flow” which enables advanced static analysis techniques like data-flow analysis to be explored. Oink is also open source, just like Trike.

Other Static Analysis tools that exist in industry were not explored because of their high licensing cost and because many of the useful comparisons like runtime,

the number of false positives and coverage are not yet relevant because Trike is a prototype and currently limited to structural static analysis.

2.2 Relational Databases

A relational database enables developers to express relationships between sets of data. Mathematical set theory provides the basis for building these relationships while the SQL language provides the means for conveying them in a structured and consistent manner [9]. These constructs provide the foundation for the logic associated with the storage, modification and retrieval of arbitrarily large amounts of data [9]. Such data are stored in tables, with each row of a table containing a unique record comprised of the data needed to describe the entity data stored in the row, partitioned into columns. Each row is identified with a unique id “(known as the *primary key*)” that helps distinguish it from other entities in the table. The *primary key* forms the foundation for building relationships across tables relating entities to other entities [2].

The tables used to store the data in a database are created using SQL schema statements, data is retrieved and modified in the database using SQL data statements and transactions are initiated, committed and rolled back using SQL transaction statements. Many implementations of these technologies exist but the one I used is a branch of the relational database MySQL called MariaDB. “MariaDB is a database server that offers drop-in replacement functionality for MySQL.” [55] It is a relative newcomer to the field of relational databases but is based on the very well established code base of MySQL. MariaDB strives to maintain compatibility with MySQL, now owned by Oracle Corporation [41], but also seeks to implement new and interesting features useful to developers.

2.3 Extensible Markup Language

The initial purpose behind the development of Extensible Markup Language (XML) was to create a portable platform agnostic representation of structured data [6,12,51].

XML is now an established technology used by organizations around the world [51]. Beyond this original hierarchical design of variably nested empty and non-empty elements, XML enables developers to construct their own rule-sets referred to as schemas, that enforce the structure of data stored in XML files. Established tools like XPath help developers traverse documents using root nodes and more recent tools like XQuery enable developers to combine XML documents with other documents to derive new output [52, 53]. XML even has its own transformation language called XSL Transformations (XSLT) that empowers developers to change XML documents into other types of XML documents such as HTML, XHTML or SVG as well as add presentation enhancements [54]. The real power behind XML with regard to Static Analysis is that it provides an implicit model for explaining the relationship between source code elements. GCC-XML generates XML documents using the intermediate representation of C++ source code and as a result provides Trike with the implicit source element relationships of a project.

CHAPTER III

Trike Architecture

Trike is the name given to the code performing the work outlined in this Thesis. The name “Trike” is a colloquialism for the word tricycle and was chosen because many people first ride a tricycle as a child. It is hoped that Trike will serve a similar purpose with regard to *Static Analysis*. Using a new approach to established techniques, developers should be able to use Trike to improve their C++ code base. Trike is written in the Python language and provides a scaffolding around the individual components used in this Thesis, such as GCC-XML and MariaDB, to create a cohesive and usable tool from disparate components.

Trike generates XML using GCC-XML so the source code it analyzes needs to be capable of being built by the GCC tool-chain. Trike provides the logic and data to GCC-XML required to generate XML representation of source files. The provided logic detects source files using the extensions specified in the Trike configuration file with a simple pattern matching algorithm. Once the list of valid files is generated, Trike then calls GCC-XML for each file from the list and generates an XML representation of the source based upon the intermediate representation created by GCC. The scaffolding Trike provides to MariaDB consists of the logic used to process the schema definition file specified in the Trike configuration file, logic to apply the statements to the database and logic to ultimately generate and run queries requested by

other portions of the Trike code base. Trike also provides the logic for generating queries to upload the entire contents of the original source files and their generated XML files, provides the logic for parsing the generated XML using the Simple API for XML (SAX) and includes the logic for loading the generated data into the database. In the future, Trike will also provides a utility to regenerate XML representations of modified source files and replace their data in the database.

3.1 The GCC-XML Extension

The GNU Compiler Collection (GCC) implements multiple front-ends in its main-line source tree used to transform raw source code into abstract trees which can be transformed, through lower level processes, into machine code and ultimately into an executable [49]. The C++ front-end to GCC implements ISO C++ and is able to parse whole source trees [49]. The purpose of the C++ front-end is to generate consumables for other parts of the compilation process [49]. These efforts are naturally not geared towards generating alternative representations of source code. For instance, if a developer wanted to perform whole C++ program parsing and generate a program representation in extensible Markup Language (XML) they would have to create a wrapper, or quite possibly an extension to take the intermediate representation of the parsed C++ source and generate XML.

This is not a trivial process, which is quite possibly why few tools exist to perform this function [24,40]. GCC-XML is one of the few tools that can generate XML from C++ source code. GCC-XML leverages the C++ front-end to GCC which is important as GCC is able to deal with the entire C++ language specification [24]. It is important to make note that GCC-XML is oriented to generating XML representations for each individual source file. This means that when it generates XML for each individual source file, it is prone to generating redundant information. Although, this is ideal for development tools like Integrated Development Environments (IDE) which need to convey source code in a tangible and useful way to developers on a per file basis, it is not ideal for tools like Trike. Development environments often make

use of visual elements to enable developers to collapse and expand functions to save space on a page and enhance understanding of a code base, to highlight variables, types and other syntactically relevant elements in different colors to deepen understanding of the code displayed on the screen, while Trike is interested in generating whole program representations using non-redundant elements. This augmentation to Trike is still under development.

Despite this fleeting shortcoming, the utility of GCC-XML did not escape me, as it makes whole program representation easy and partial program analysis accessible. Before GCC-XML generates XML it creates an intermediate representation with all includes resolved and imported. Once this step is complete it generates the XML representation of the code and saves it as specified by the user. Trike calls GCC-XML for each valid source file and saves the generated XML using the original filename but with an XML extension instead of the original extension. An alternative approach would have been to generate the intermediate representation of the source code for the entire program, save it, then call GCC-XML on this file, which in the long run might lead to less elemental redundancy in the database. This is especially true with respect to maintaining proper namespace for portions of code. It is acknowledged that the whole intermediate approach might have been superior to the one chosen but the current approach will enable Trike to selectively regenerate XML representations which when coupled with logic to detect existing elements should enable fast and efficient partial program analysis after developers make minor changes.

While building Trike, several assumptions were made. The most important assumption, which was verified before running the tool, was that the source code given to Trike builds with few warnings and no errors. Another was that the project contained easily discoverable include directories where header files and other important portions of code were accessible. The last, and most important, assumption was that GCC-XML was bug free to the point of being usable on a very large scale.

The last assumption merits further discussion as it became apparent that when analyzing large projects GCC-XML works to the point the code compiles. This operation can take quite a long time which on slower machines can present the user

with no way to monitor the progress or track errors. I also suspect GCC-XML could be improved by adding logic to capture important components of the code base like function bodies which would enhance the analysis capabilities of Trike. To accommodate generated warnings, Trike includes logging to a database table as well as logging of all operations to a log file. For longer builds, Trike also includes a progress indicator to help indicate the progression throughout the XML generation process. If the generation process is interrupted, by the user or by an error, the project must be deleted, and all data reprocessed. Future releases should make this scenario less painful by relying on the innate capability of GCC to selectively generate representations of files that are missing or modified.

The progress indicator and logging will prove useful in future work when we begin to compare Trike to competing *Static Analysis* tools. The file counter currently indicates how many files have generated out of the total and the log currently includes time-stamps so the user can begin to estimate how long each run should take. These elements are essential for developers working under time constraints as they will help them appropriately schedule their builds. Trike also indicates which files failed to generate and provides the user with a list in addition to the STDERR contents in a database table. An added benefit of cataloging these messages is that some of them might correspond to valid compiler failures. Even if they represent false positives they are cataloged and are not presented in a manner that would confuse developers or distract them from their main purpose. With tools like Trike, having false positives is often better than false negatives [16]. I believe the current logging approach balances utility with the responsibility of reporting potential errors that in most environments should be appreciated. I also believe logging in Trike should be more granular but that the current level of logging is commensurate with the current capabilities of Trike.

3.2 The XML Schema

The Document Type Definition (DTD) contributed by the GCC-XML developers was used to determine what language elements were currently mapped from C++ to XML by GCC-XML [21–23]. The rules contained in the definition were used to ascertain which elements required which attributes and to a limited extent what type of data each attribute should contain. I am aware the language elements mapped in the DTD might not contain every possible aspect of the C++ grammar because GCC-XML is based on GCC. I am also aware the current DTD is an amalgam of efforts by volunteer contributors to GCC-XML and that the authors of GCC-XML acknowledge it is still a work in progress [21–23].

Today, GCC-XML supports many useful components of the C++ language. The language elements and related meta-data elements that are currently mapped include argument, arraytype, base, class, constructor, converter, cvqualifiedtype, destructor, enumeration, enumvalue, field, file, function, functiontype, fundamental-type, method, methodtype, namespace, namespacealias, offsettype, operatorfunction, operatormethod, pointertype, referencetype, struct, typedef, unimplemented, union, and variable. I believe unimplemented serves as a “catch-all“ for elements not currently supported by GCC-XML. Because GCC-XML is still under development, the XML schema was not used to actively validate the generated XML. However, with some changes and further inspection of GCC-XML, using the schema for such verification could prove useful to ensure the integrity of generated data to a particular C++ dialect and might lead to a beneficial comparison to existing schema query tools [61].

3.3 The Database Schema

The database holds all of the data generated by Trike with the exception of the data generated by the log file and the settings stored in the configuration file. The underlying database schema is essential to the organization of this data and a logical requirement for building relationships about various aspects of the code base to realize

the utility of Trike as a *Static Analysis* tool. In order for developers to harness the data generated by Trike, each XML element needs to be properly mapped to a table and the element attributes to columns in the table. Work on mapping the hierarchy of elements (and their attributes) to a database exists in the literature and was used as the basis for my approach [6,12,17,18,31,47,50]. In total, 30 tables exist that map the 30 XML elements present in the XML files generated by GCC-XML. Each table in the schema is named with a prefix of **gccxml_** then the name of the element. The one exception to this is the table for the root element which is simply named **gccxml**. The additional five tables that exist in the database are for storing useful meta-data about the project such as **errors** table for holding GCC-XML errors related to the project, the **program** table for holding meta-data about the project, the **file** table for holding meta-data about source and XML files, the **source** table for holding a reference to a valid source file in the **file** table and the actual source from the file, and finally the ‘xml’ table for holding a reference to a valid xml file in the ‘file’ table and the actual xml from the generated xml file. The following graphic shows the relationships between some of the tables in the database for the C++ Sockets Library project and was generated using SchemaSpy [44]. An alternative, and complete listing of tables is shown in 2 the “Show Tables” figure after the partial representation shown in 1.

Figure 1: Example Trike Database Relationships

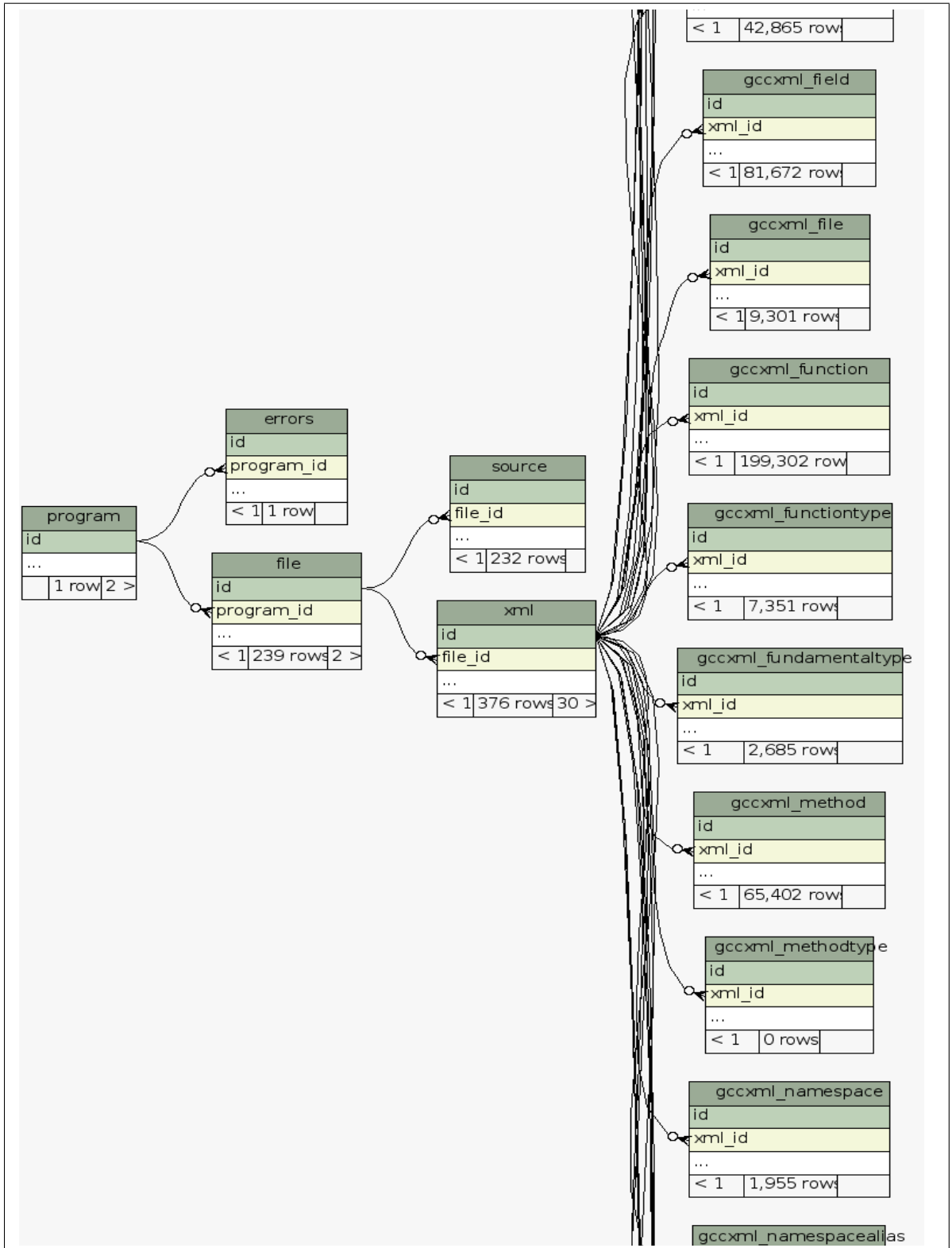


Figure 2: Table Listing of Trike Database Schema

```
MariaDB [trike]> show tables;
+-----+
| Tables_in_trike |
+-----+
| errors           |
| file            |
| gccxml          |
| gccxml_argument |
| gccxml_arraytype |
| gccxml_base     |
| gccxml_class    |
| gccxml_constructor |
| gccxml_converter |
| gccxml_cvqualifiedtype |
| gccxml_destructor |
| gccxml_enumeration |
| gccxml_enumvalue |
| gccxml_field    |
| gccxml_file     |
| gccxml_function |
| gccxml_functiontype |
| gccxml_fundamentaltype |
| gccxml_method   |
| gccxml_methodtype |
| gccxml_namespace |
| gccxml_namespacealias |
| gccxml_offsettype |
| gccxml_operatorfunction |
| gccxml_operatormethod |
| gccxml_pointertype |
| gccxml_referencetype |
| gccxml_struct   |
| gccxml_typedef  |
| gccxml_unimplemented |
| gccxml_union    |
| gccxml_variable |
| program        |
| source         |
| xml            |
+-----+
35 rows in set (0.00 sec)
```

3.3.1 Mapping XML to SQL DDL Statements

GCC-XML generates non-empty XML elements containing other empty and/or non-empty elements containing an arbitrarily complex nesting of other elements. This layout requires an element type to table row mapping with a column for each attribute of the element. This element to table row and attribute to column mapping captures the structural aspects of the code base which enables foreign key references to be used to create implicit relationships between pertinent elements. The type of data each attribute contains dictates the type for the corresponding column in the table for the element. For example, if the attribute has a limited set of valid values such as a 0 or a 1, or of “public,” “private,” or “protected,” an enum was used. If the attribute values are of an unpredictable length and comprised of numbers and characters then text is used. If the attribute values are of predictable length and comprised of numbers and characters then varchar with an appropriate length is used. I acknowledge such an approach might seem arbitrary but in practice it works quite well. A more exhaustive approach to mapping types between SQL and XML is explored in the literature [17, 18, 31, 47, 50]. The minor tradeoff with my naïve approach of mapping XML elements to SQL Data Definition Language (DDL) statements is all entities are treated as sequences of characters. No special considerations are made to treat characters as character data CDATA or parsed character data PCDATA.

CHAPTER IV

Static Analysis with Trike

Trike is a different Static Analysis tool. Trike leverages the power of a relational database to structure code relationships and leverages SQL to provide developers with a method to create their own, unique data extraction queries. The usage of SQL alleviates the burden of learning a restricting tool specific language. Although the ability to write extensions for a Static Analysis tool is not novel, writing such queries in a ubiquitous language like SQL, is to my knowledge, a new approach. To illustrate the usefulness of Trike and the power of using a relational database and SQL, I chose three C++ projects and wrote structural queries to learn about each project. I then wrote composite queries, using the results of the structural queries to learn even more about each project, and used the knowledge I gained by developing these initial queries to investigate how to begin to write more advanced queries that might help identify errors in these projects and other C++ code bases. The queries I present and discuss in this chapter should prove useful to other developers that want to learn more about these projects as well as their own code. I conclude this chapter with a discussion of queries I would like to see Trike perform but currently cannot due to various limitations imposed upon Trike by GCC-XML 4.3.

4.1 C++ Projects Analyzed

Analyzing the code of an established project helps a developer learn about code design and structure before attempting to introduce changes. Reading source code is one way to analyze code and can lead to a concrete understanding of architecture and implementation. Meta-data can enhance a developers understanding of source code by providing context. In particular, meta-data can help a developer understand the style and preferences of existing developers without having to analyze large code blocks on their own. Preferences like the ratio of public, private and protected data members for classes in the code base might indicate how disciplined developers are at data encapsulation while others like a small average function length might indicate a proclivity to create well defined, reusable, purpose driven pieces of code; large functions might indicate developer immaturity, code obfuscation and a poorly designed code base. Some of these meta-data can be gathered through reading while others cannot. Regardless, it is often easier to have a tool extract it for you, which Trike does quite readily.

Other structural data like the number of Source Lines of Code (SLOC) for the entire project, SLOC per file, average SLOC per class, the number of header files, the number of source files and the number of library files help reveal other important architectural choices. I extract these meta-data and a few others using Trike for the libodc++ library, the Apache log4cxx library and the C++ sockets library and discuss their relevance in the “Existing Trike Queries” 4.2 section. This section is divided into three sub-sections. The first 4.2.1 sub-section describes structural queries 4.2.1, the second describes composite queries derived from structural meta-data 4.2.2 and the third describes progress on developing advanced queries that could be used as the basis for detecting errors 4.2.3. After I introduce a query I describe the thought process behind the development of the query. Initial results seem promising and are provided at the end of the first two sub-sections and full results are presented in the appendices 5.2. Before I present these queries and the initial output it will prove helpful to discuss the importance of the projects I chose and elaborate on why I chose

to use them as examples.

The libodbc++ Library

Open Database Connectivity (ODBC) software provides software developers with a standardized Application Programming Interface (API) for accessing information in a relational database []. The libodbc++ project uses the API provided by ODBC to build a useful database access library for C++ developers. The project has the single goal of “creat[ing] a complete c++ development environment for SQL database access in [the] form of a class library and a set of tools.” The website for the project expands upon this definition and states the following:

libodbc++ is a c++ class library for accessing SQL databases. It is designed with standards in mind, so it provides a subset of the well-known JDBC 2.0(tm) and runs on top of ODBC. It is distributed under the LGPL. The library has shown to compile and work on Linux, FreeBSD and win32 platforms. Any environment with a reasonable compiler, standard c++ library and ODBC Driver Manager should do. In general, the library should run on anything that has a reasonable c++ environment and an ODBC driver manager version 2.5 or higher.

Developers that use libodbc++ in their C++ projects use it to retrieve data from a relational database. As the core of a programs database logic, libodc++ is one of the most important parts of a C++ program. This is why I chose to analyze the libodc++ project with Trike.

The Apache log4cxx Library

Logging application data is important. Log data can provide insights into user interaction, security, debugging, and other important information. The Apache Software Foundation (ASF) describes the Apache log4cxx library “[a]s a logging

framework for C++ patterned after Apache log4j.” Reserved for applications utilizing the Apache Portable Runtime (APR) the Apache log4cxx project is useful for many projects. It is a liberally licensed project, licensed under the Apache License, “is designed to be reliable, fast and extensible ...” and “... strives to be simple to understand and [easy] to use.” I chose to analyze log4cxx with Trike for two reasons: 1) log4cxx is used by a large number of C++ projects, including the Apache Web Server and 2) logging is vital to helping developers discover security issues, especially issues related to unanticipated usage scenarios.

The C++ Sockets Library

Sockets are essential to building network enabled applications. The C++ Sockets Library “is a GPL licensed C++ class library wrapping the Berkeley sockets C API, and therefore works on most unices and also win32. The library is in use in a number of real world applications, both commercial and open source.” The C++ Sockets Library provides support for the Secure Sockets Layer (SSL), addressing network connected systems using Internet Protocol version 4 (IPv4) and version 6 (IPv6), constructing TCP, UDP and SCTP sockets, and communicating using the HTTP(S) protocol. The library also provides a highly customizable error handling framework and claims to work on Linux and most versions of Windows. It also claims to have partial support for Solaris and Mac OS X. I chose to evaluate the C++ Sockets Library with Trike because it provides C++ developers with an essential piece of code for accessing the network and claims to work on a variety of platforms.

4.2 Existing Trike Queries

SQL queries used to build relationships for data stored in Trike are similar to conventional SQL queries run against data contained in a relational database. Each row in a table, has a column containing a unique key, that distinguishes it from other data in the table. These keys can be used to build relationships with rows contained

in other tables. Some tables contain multiple columns that enable this type of relationship building. Because Trike loads data for each discovered element into the rows of a table, this type of relationship building provides the basis for modeling element interactions in a code base.

I first investigated writing structural queries to learn more about the composition of libodbc++, log4cxx and the C++ Sockets Library. I used the common UNIX utility *wc* to verify the results of some of the queries I developed and when appropriate indicated if the output matched the results in the output tables 4.2.1 4.2.2. I used a hyphen (-) to indicated a query did not have a *wc* equivalent. These checks were important, because *wc* is an established programming tool relied upon many developers to quickly gather structural data on files. They were also important because *wc* is restricted to UNIX and UNIX-like platforms while Trike enables developers on many different platforms to gather the same structural data. Trike is intrinsically cross-platform and supported on the same platforms as MariaDB which include Linux, Solaris, Windows and Mac OS X.

4.2.1 Structural Queries

The first set of queries I developed using Trike were structural queries. The output for most of these queries is contained in a table at the end of this sub-section 4.2.1 and the data which could not be contained in the table is located in the appropriate appendix A-1. When possible and appropriate I indicated in the output table whether the output gathered using *wc* matched the output gathered using my query. In rare instances the data were slightly different. I attributed these discrepancies to documented compilation issues which were captured in the **errors** table or to the duplicate data generated by GCC-XML and subsequently inserted into the database. I attempted to minimize the impact of the latter by utilizing the **distinct()** operator and joins when possible.

When I started developing structural queries for Trike I started at what I believed to be the most logical point, determining how many files were in a project. The SQL for this query is shown in the following block.

The Number of Files in a Program

```
1 MariaDB [trike]> select count(id) as "Number of Files"
2                   from file
3                   where type = "source";
```

When I developed this query I utilized the `count()` operator to determine the number of file entities contained in the `file` table. The `file` table serves as the registry for all files in the project. Each entity contained in the `file` table has a *type* which can be “source” or “xml,” a *program_id* which corresponds to the unique project identifier for the entity contained in the `program` table, a *name* which contains the short name of the file and a *path* which contains the fully qualified path to the file on the filesystem. The *primary key* for each entity contained in the `file` table is used to build relationships with the generated XML table named `xml` and a few other tables.

After I developed a query to determine the number of files in the project, I then thought about existing Static Analysis tools and realized that the next structural aspect developers would find useful would be determining where code tends to gravitate. The answer to this question involves several queries. Each query helps determine if the majority of code is contained in header files, standard source files or system/library includes. To start to answer this question I first developed a SQL query to determine the number of header files. The SQL query used to determine this is shown in the following block.

The Number of Header Files in a Program

```
1 MariaDB [trike]> select count(id) as "Number of Header Files"
2                   from file
3                   where type = "source"
4                   and name like "%.h";
```

This query also uses the `file` table and the `count()` operator. I made it more specific by selecting the file type as well as the file extension. In this case the appropriate file extensions were `.h` or `.hpp`. I used `.h` because the projects I analyzed did not contain `.hpp` files.

The next SQL query I wrote helped determine the number of source files present in the code base. It is very similar to the previous query, with the one exception of file

extension. Instead of using *.h* or *.hpp* I used *.cpp*. The SQL query used to determine this number is shown in the following block.

The Number of Source Files in a Program

```
1 MariaDB [trike]> select count(id) as "Number of Source Files"
2
3           from file
4           where type = "source"
5           and name like "%.cpp";
```

I then thought about code locality beyond file counts and wondered what kind of query could be constructed to determine the Source Lines of Code (SLOC) for the entire program, and then for each file in the program. The SQL for the determining the SLOC for the entire program is shown in the following block.

The Total Lines in a Program (SLOC)

```
1 MariaDB [trike]> select
2           sum(
3             (
4               length(source.contents)
5             -
6               length(replace(source.contents, "\n", ""))
7             )
8           )
9           as "SLOC"
10          from source, file
11          where source.file_id = file.id;
```

The SLOC query uses the **source** table which contains a *file_id* column referencing the appropriate entity from the **file** table and a *contents* column used to store the entire contents of the referenced file. The SLOC calculation is performed in the database using the **sum()** and **length()** operators.

I then thought about the SLOC for a program in terms of my original question about determining the number of files in the project. This led me to try to determine how many lines of code were in each individual file. The SQL query for determining the SLOC for each file is shown in the following block. The results for this query are included in the appendix A-1.

The SLOC per File

```
1 MariaDB [trike]> select file.id,file.name ,
2           length(source.contents)
3           -
4           length(replace(source.contents ,"\n",""))
5           as "SLOC"
6 from file,source
7       where file.type = "source"
8           and file.id = source.file_id;
```

I then thought about how these numbers could be used to tell me how many lines of code were in header files and how many lines of code were in source files. These data would complement the previous queries for determining the number of header files and source files by creating a clearer picture of code locality. The first SQL query, the one for determining the SLOC in header files is shown in the following block. The results for this query are included in the appendix A-1.

The SLOC for all Header Files

```
1 MariaDB [trike]> select
2           sum(
3           length(source.contents)
4           -
5           length(replace(source.contents ,"\n",""))
6           )
7           as "The SLOC in Header Files"
8 from file,source
9       where file.type = "source"
10            and file.id = source.file_id
11            and file.name like "%.h";
```

This query is similar to the query for determining the SLOC for a program but excludes files which are not header files by filtering files based on their extension. In this case I used *.h* to create a filter for header files.

The second SQL query, the one for determining the SLOC in source files is shown in the following block and is quite similar to the one used to determine SLOC for each file. Instead of using *.h* or *.hpp* I used *.cpp* to select source files. The results for this query are included in the appendix A-1.

The SLOC for all Source Files

```
1 MariaDB [trike]> select
2     sum(
3         length(source.contents)
4         -
5         length(replace(source.contents, "\n", ""))
6     )
7     as "The SLOC in Source Files"
8 from file, source
9     where file.type = "source"
10         and file.id = source.file_id
11         and file.name like "%.cpp";
```

I then wanted to know how many bytes were contained in each of the files to help estimate source size. This query does not introduce anything that has not already been utilized in other queries but instead of generating a difference between file contents with newlines characters and without newlines characters we simply count the bytes in the source file. The SQL query for the byte count is shown in the following block and the results are included in the appendix A-1.

The Bytes per File

```
1 MariaDB [trike]> select file.name, length(source.contents)
2     as "Byte Count"
3 from source, file
4     where source.file_id = file.id;
```

As an artifact of the number of bytes used to store characters on my platform, the byte count also corresponded to the number of characters contained in each file. This would not be true for locales requiring more than one byte to represent a character but could be easily adjusted to accommodate different locales if a developer wanted to use this query to calculate the number of characters in a file written in their native language.

After determining the size of files in the code base I wanted to improve my understanding of file contents and code locality. The first query I explored was calculating the number of functions in the entire code base including system and library files. The SQL query for this calculation is shown in the following block.

The Number of Functions in a Program

```
1 MariaDB [trike]> select count(distinct(name))
2                   as "Function Count"
3                   from gccxml_function;
```

This is the first query to utilize the **distinct()** function to eliminate duplicates generated by GCC-XML and inserted into the GCC-XML function table named **gccxml_function**. I then realized that it would be helpful to determine the number of functions contained in each generated XML file. The SQL query for this calculation is shown in the following block and is similar to the previous query but I wanted to see if I could elicit a per file function count using the **gccxml_function** table and the **gccxml_file** table. It proved difficult to use the raw source for this calculation, due to the structure of the data generated by GCC-XML, but it was possible using the generated xml file contents. The results for this query are included in the appendix A-1.

The Number of Functions per Generated XML File

```
1 MariaDB [trike]> select gccxml_file.xml_id, count(distinct(gccxml_function.id))
2                   as "Function Count per XML File"
3                   from gccxml_function, gccxml_file
4                   where gccxml_function.xml_id = gccxml_file.xml_id
5                   group by gccxml_file.xml_id;
```

I then wanted to count the number of distinct classes in the code base. The SQL query for calculating this number is shown in the following block. This query is very similar to the function count query only instead of using the **gccxml_function** table it uses the **gccxml_class** table.

Total Program Class Count

```
1 MariaDB [trike]> select count(distinct(name))
2                   as "Class Count"
3                   from gccxml_class;
```

After calculating the number of functions, the number of functions per file, and the number of classes in each program, I then wanted to count the number of distinct variables in the code base as this could help me prepare tests for evaluating

the code. The more variables that exist in the code base, the more states I would have to reach and the more tests I would have to write to thoroughly evaluate the code base. The query I developed utilizes methods introduced in previous queries but used the **gccxml_variable** table. The SQL query for calculating this number is shown in the following block.

Total Variable Count

```
1 MariaDB [trike]> select count(distinct(name))
2                   as "Variable Count"
3                   from gccxml_variable;
```

After calculating the total number of variables in the code base I then thought it would be useful to divide the number into private, protected and public class variables contained in the code base. The query I developed use the same approach as the “Total Variable Count” query but utilizes the **access** column for entities stored in the **gccxml_variable** table. Valid values for the “access” column include “private,” “protected,” or “public.” Determining the number of each of these variables should help me learn more about how well or how poorly data is encapsulated throughout the code base as well as help with testing. The SQL query to calculate the number of private variables is shown in the following block and can easily be changed to calculate protected and public variables by adjusting the value of the “access” parameter as shown in the two subsequent queries.

Total Private Variable Count

```
1 MariaDB [trike]> select count(distinct(name))
2                   as "Private Variable Count"
3                   from gccxml_variable
4                   where access = "private";
```

Total Protected Variable Count

```
1 MariaDB [trike]> select count(distinct(name))
2                   as "Protected Variable Count"
3                   from gccxml_variable
4                   where access = "protected";
```

Total Public Variable Count

```
1 MariaDB [trike]> select count(distinct(name))
2                   as "Public Variable Count"
3                   from gccxml_variable
4                   where access = "public";
```

Now that I knew more about each project I wanted to start learning more about the code each of these projects depended on. The first question I had was what system level and library files were included by GCC while generating the intermediate representation. The SQL query for generating this list of dependencies is shown in the following block.

List of Included Library Files

```
1 MariaDB [trike]> select distinct(gccxml_file.name)
2                   from gccxml_file, xml
3                   where xml.id in
4                   (
5                       select distinct(gccxml_file.xml_id)
6                           from gccxml_file
7                   )
8                   and gccxml_file.fid not in
9                   (select id from file);
```

This query uses previous methods, like **distinct()** to remove duplicate data, but also introduces the concept of sub-selects where I generate a list of files and use those lists to filter potential candidates from the final result set. The results of this query are included in the appendix A-1.

I then wanted to calculate the number of system level and library dependencies included by GCC. This should help later when I seek to determine the percentage of code reuse attributable to code contained outside of the code base versus code contained inside of the code base. The SQL query for determining this number is shown in the following block. It uses methods and tables introduced through previous queries.

The Number of Included Library Files

```
1 MariaDB [trike]> select count(distinct(gccxml_file.name))
2                   as "Number of Included Library Files"
3                   from gccxml_file, xml
4                   where xml.id in
5                       (select distinct gccxml_file.xml_id from gccxml_file)
6                       and gccxml_file.fid not in
7                       (select id from file);
```

I then wanted to determine the particular importance of a file in the source base. The level of importance is determined by calculating the number of times a particular file is referenced in other files as an include. The SQL query for determining these numbers for all files in the project code base is shown below and the results are shown in the appendix A-1.

File Weight

```
1 MariaDB [trike]> select file.id,file.name ,
2                   (select count(gccxml_file.id)
3                   from gccxml_file
4                   where file.id = gccxml_file.fid)
5                   as "File Weight"
6                   from file
7                   where file.type = "source";
```

The results from this query revealed something interesting: many files had a zero file weight. Further investigation warranted, I looked at the source for a small sample of files and verified that indeed the files with a zero file weight were not included in any other source files in the project code base. Some were included in tests but test directories were excluded from the discovery process. This data is priceless when a developer wants to determine if they can safely remove a file from code and in situations when they need to determine if they need to test the code contained in a file or are trying to localize an issue and need to know if they should check particular files or not. These results might also indicate the code is simply standalone code. In either case it still gives the developer context when analyzing the code base.

If a developer wants to determine the file weight for a particular file they first

need to determine the unique id for the file in the **file** table. The SQL query to determine the id for the “Socket.cpp” file in the C++ Sockets Library is shown below along with the output.

Get the File ID for a Particular File

```
1 MariaDB [trike]> select id
2                   as "FID"
3                   from file
4                   where name = "Socket.cpp"
5                   and type = "source";
```

The File ID for the Socket.cpp file in the C++ Sockets Library

```
1 +-----+
2 | FID |
3 +-----+
4 | 114 |
5 +-----+
6 1 row in set (0.00 sec)
```

After the developer determines this number they then substitute this number for the text marked “FID” in the SQL query shown below to determine the file weight.

Get the File Weight for a Particular File

```
1 MariaDB [trike]> select count(gccxml_file.id)
2                   as "File Weight"
3                   from gccxml_file
4                   where fid = "FID";
```

There were many queries discussed in this sub-section. I first started with a query to determine the number of files contained in a program. I then asked about code locality, where code is written and new functionality is typically added. These questions were answered in part by developing queries to tell me how many header files, source files, and system and library files were in a program. I then started to ask more questions about the code base such as how many functions, classes and variables existed in the code base. Like previous questions, initial queries were developed to begin to answer these questions. Finally, I wrote queries to help me develop an understanding of how data is (or is not) encapsulated in the project and what code is

<i>Query</i>	<i>libodbc++</i>	<i>log4cxx</i>	<i>sockets</i>	<i>wc matches?</i>
The Number of Files	42	325	122	-
The Number of Header Files	23	177	67	-
The Number of Source Files	19	148	55	-
Total Source Lines of Code (SLOC)	16444	45733	26869	Y
Total Header SLOC	6408	22356	8310	Y
Total Source SLOC	10036	23377	18559	Y
The Number of Functions	1298	1680	3331	-
Total Class Count	361	752	280	-
Total Variable Count	392	1050	285	-
Total Private Variable Count	81	23	40	-
Total Protected Variable Count	0	0	1	-
Total Public Variable Count	311	1029	244	-
Number of Included Files	170	159	139	-

Table I: Structural Query Output

actually used in the project. Each query added to my understanding of the code base for each project by providing a distinct piece of meta-data as shown in the table at the end of this sub-section. Each piece of meta-data proved useful on its own and as I will soon show will be made even more useful when combined with other meta-data 4.2.2.

4.2.2 Composite Queries

After developing the structural queries in the previous section 4.2.1, I began to think about how the results could be combined to build a better picture of the code base. The first question I asked combined the SLOC for each file and the number of files to derive the average SLOC per file. This number is important to determine the average SLOC of the files contained in the code base. A large number of files coupled with a low average SLOC or a small number of files coupled with a high average SLOC might indicate poor design. The SQL query for this calculation is shown below and utilizes “The Total Lines in a Program (SLOC)” query and “The Number of Source Files in a Program” query.

Average SLOC for All Files

```
1 MariaDB [trike]> select
2     sum(
3     (
4     length(source.contents)
5     -
6     length(replace(source.contents, "\n", ""))
7     )
8     )
9     /
10    count(file.id)
11    as "Average SLOC"
12  from file, source
13  where file.type = "source"
14        and file.id = source.file_id;
```

I then asked the same question for just header files. The SQL query for this calculation is shown below and uses the “The SLOC for all Header Files” query and “The Number of Header Files in a Program” query.

Average SLOC for Header Files

```
1 MariaDB [trike]> select sum(
2     (
3     length(source.contents)
4     -
5     length(replace(source.contents, "\n", ""))
6     )
7     )
8     /
9     count(file.id)
10    as "Average .h SLOC"
11  from file, source
12  where file.type = "source"
13        and file.id = source.file_id
14        and file.name like "%.h";
```

I then asked the same question for just source files. The SQL query for this calculation is shown below and uses the “The SLOC for all Source Files” query and “The Number of Source Files in a Program” query.

Average SLOC for Source Files

```
1 MariaDB [trike]> select sum(  
2     (  
3         length(source.contents)  
4         -  
5         length(replace(source.contents, "\n", ""))  
6     )  
7 )  
8 /  
9     count(file.id)  
10     as "Average .cpp SLOC"  
11 from file, source  
12     where file.type = "source"  
13           and file.id = source.file_id  
14           and file.name like "%.cpp";
```

After calculating the average SLOC for the entire program and then for header and source files, I then returned to a question I raised in the previous section regarding the density of variables with particular access 4.2. To calculate the proportion of private, protected and public variables in the code base I combined the queries I used to calculate them individually into one query using the `concat()` operator. The query for calculating this proportion is shown below.

Proportion of Private, Protected and Public Variables

```
1 MariaDB [trike]> select  
2     concat(  
3         (select count(distinct(name))  
4           from gccxml_variable  
5           where access = private),  
6         to ,  
7         (select count(distinct(name))  
8           from gccxml_variable  
9           where access = protected),  
10        to ,  
11        (select count(distinct(name))  
12          from gccxml_variable  
13          where access = public)  
14     )  
15     as Private to Protected to Public;
```

We can also calculate the percentage of the total variables for each level of visibility compared to the total number of variables in the program. This percentage

is another representation of the data the proportion data reveals and can also help estimate how well data might be encapsulated throughout the code base. The SQL query for calculating the percentage of “private” variables is shown below with a subsequent queries for “protected” and “public” variables.

Percentage Private Variables

```
1 MariaDB [trike]> select
2             concat(
3                 ((select count(distinct(name))
4                    from gccxml_variable
5                    where access = "private")
6                 /
7                 (select count(distinct(name))
8                    from gccxml_variable))
9             * 100,)as "Percentage Private Variables";
```

Percentage Protected Variables

```
1 MariaDB [trike]> select
2             concat(
3                 ((select count(distinct(name))
4                    from gccxml_variable
5                    where access = "protected")
6                 /
7                 (select count(distinct(name))
8                    from gccxml_variable))
9             * 100,)as "Percentage Protected Variables";
```

Percentage Public Variables

```
1 MariaDB [trike]> select
2             concat(
3                 ((select count(distinct(name))
4                    from gccxml_variable
5                    where access = "public")
6                 /
7                 (select count(distinct(name))
8                    from gccxml_variable))
9             * 100,)as "Percentage Public Variables";
```

After I explored the visibility of variables in the program I then investigated developing a query to try and determine which percentage of the code base was actually being used to generate the program. This idea came about after I made the

discovery in the previous section 4.2 that some files were not referenced by any other file. This query proved elusive and in order to calculate the actual percentage of code being used in the code base I needed to develop a set of three queries. The first query returned all files that are referenced as includes. The second query retrieved the number of rows in the first result and the third query calculated the percentage of files referenced as includes over the total number of files in the project. The set of SQL queries are shown below and the results are included in the table.

Set of Queries to Calculate Percentage of Files Referenced

```
1  -- QUERY #1
2  MariaDB [trike]> select (
3      select count(gccxml_file.id)
4      from gccxml_file
5      where file.id = gccxml_file.fid
6  )
7      as "File_Weight"
8  from file
9      where file.type = "source"
10     group by file.id
11     HAVING File_Weight > 0;
12
13 -- QUERY #2
14 MariaDB [trike]> select found_rows()
15     as "Number Files With Dependencies";
16
17 -- QUERY #3
18 MariaDB [trike]> select concat(
19     ("QUERY #2 OUTPUT"/"TOTAL FILES") * 100
20     ,"%")
21     as "Percentage of Files Referenced"
22 from dual;
```

The final query I developed was to calculate the percentage of code reuse measured by the number of system and library included from outside of the native code base over the total number of files used to build the final executable. The SQL query to calculate this number is shown below and uses a variety of concepts and

<i>Query</i>	<i>libdbc ++</i>	<i>log4cxx</i>	<i>sockets</i>	<i>wc matches?</i>
Average SLOC	391.5238	140.7169	220.2377	Y
Average Header SLOC	278.6087	126.3051	124.0299	Y
Average Source SLOC	528.2105	157.9527	337.4364	Y
Percentage Private Variables	20.6633	2.1905	14.0351	-
Percentage Protected Variables	0.0000	0.0000	0.3509	-
Percentage Public Variables	79.3367	98.0000	85.6140	-
Percentage Files Referenced	71.4286	97.8462	68.0328	-

Table II: Composite Query Output

tables introduced through previous queries. The results are shown in the appendix A-2.

Percent Code Reused

```

1 MariaDB [trike]> select
2             concat(
3             (
4             (select count(distinct(gccxml_file.name))
5             from gccxml_file,xml
6             where xml.id in
7             (select distinct(gccxml_file.xml_id)
8             from gccxml_file)
9             and gccxml_file.fid not in (select id from file)
10            )
11           /
12           (select count(id)
13           from file
14           where type = "source")
15           ),"% "
16          )
17          as "Percent Code Reused"
18         from dual;

```

This sub-section used some of the queries and meta-data developed in the structural query sub-section to extract more complex attributes. These meta-data are shown in the table included in this sub-section and in the appendix. The list of composite queries presented in this sub-section is by no means exhaustive but the ones I explored provided a glimpse into the many meaningful relationships that can be constructed using meta-data extracted by Trike. In the next section I discuss a few advanced queries that should provide a foundation for when I start work on

modifying Trike to identify errors in a program.

4.2.3 Advanced Queries

Now that I have established the utility of a relational database with regard to structural static program analysis, it makes perfect sense to discuss how Trike might be used in the future to identify errors. In addition to the structural and composite queries discussed in the previous sub-sections 4.2.14.2.2, the meta-data generated by Trike should enable developers to start exploring deeper relationships in their code base. The first step in this metamorphosis is to build context. Most errors detected by static analysis tools deal with order to provide context. In order to generate pre and post-conditions Trike needs to extract meaningful data before important markers like function calls and meaningful data after each marker. As it currently stands Trike is limited by the underlying power of GCC-XML. Nonetheless, I still managed to determine what a developer might be able to do to start detecting errors in their code base. The output for some of these queries is included in the appendix A-3.

The first step would be to identify a useful snippet of code to investigate. I have chosen a class member named “Data” in the Event class located in the C++ Sockets Library. Once this snippet of code is isolated the developer can then use variations of the following queries to extract data before and after the snippet of interest. This approach is by no means complete, but will be once Trike is able to import pertinent code as an element attribute. The queries required for this approach are shown below and for completeness, I have shown these queries using the raw source and the generated XML.

Search Source for a String

```
1 MariaDB [trike]> select file.name,source.id,source.file_id,
2                   locate(Event::Data() const,source.contents)
3                   as located_at_char
4                   from source,file
5                   where source.file_id=file.id;
```

Search XML for a String

```
1 MariaDB [trike]> select file.name,xml.id,xml.file_id,  
2                   locate(Event::Data() const,xml.contents)  
3                   as located_at_char  
4                   from xml,file  
5                   where xml.file_id=file.id;
```

Find Code Surrounding a String in Source

```
1 MariaDB [trike]> select file.name,source.id,source.file_id,  
2                   substr(source.contents from  
3                   locate(Event::Data() const,source.contents) - 20 for 40)  
4                   as data  
5                   from source, file  
6                   where source.file_id=file.id  
7                   limit 1;
```

Find XML Surrounding a String in XML

```
1 MariaDB [trike]> select file.name,xml.id,xml.file_id,  
2                   substr(xml.contents from  
3                   locate(Event::Data() const,xml.contents) - 20 for 40)  
4                   as data  
5                   from xml,file  
6                   where xml.file_id=file.id  
7                   limit 1;
```

Each of these queries give particular insight into the developers code base and form the basis for contextual analysis, as well as helping a developer begin to identify the various classes of errors outlined in the “Introduction” 1. I believe that with changes to GCC-XML or by using a more extensive tool like CIL, contextual analysis would become a reality in Trike and enable developers to identify errors in their code.

4.3 Further Query Discussion

In this section I introduced you to several queries you can use on your own C++ projects to extract useful meta-data. The first sub-section discussed various structural queries. Although many of the queries revealed useful meta-data, the initial set of queries I proposed could be improved. In particular, it would have been nice to

give developers a better understanding of functions and classes with respect to the source files and not the generated XML but the the data generated by GCC-XML proved limiting. It would also have been nice to not have to filter duplicates with the “distinct” function but the approach used to generate the XML with GCC-XML again proved limiting. Regardless of these meta-data should provide insight into each code base. In the second sub-section, I combined queries in the first sub-section to derive compound relationships. In addition to the average SLOC it would have been interesting to determine the median and mode for files in each code base as these values would provide additional insight into code placement. It was surprising to observe the over-reliance on public class members by each project and a study attempting to correlate security issues related to this reliance might prove interesting. It was also interesting to see the amount of code reused by each program. These numbers were not surprising as part of the power developers have nowadays is given to them through code reuse. Code reuse can reduce vulnerabilities and speed development so it is not surprising to see system libraries used so heavily. Libraries are also often required for code to work on a specific operating system, which could also be another reason why so many external library files were listed. Finally, in the third sub-section I discussed a few queries that should be vital to the future development of Trike as a competitive static analysis tool. Trike is currently very useful for performing structural static analysis but needs more work to identify errors in C++ code. Nonetheless, the results of these queries indicate the relational database should be regarded as a capable tool for performing static program analysis.

CHAPTER V

Discussion

Trike is useful for generating structural meta-data about C++ programs that can be built by GCC. Structural static analysis is useful in its own right as it enables developers to build an understanding of their code base from a compositional standpoint while enabling evolutionary structural analysis. The work I presented in this thesis represents the beginning of what one can do with a database as a Static Analysis tool. In order to access more complex aspects of Static Analysis, GCC-XML needs to be improved to expose and import particular data about the source code such as the code specific to functions and classes. It is not enough to attempt to build references to code using the raw source and elements from the database using the current version of GCC-XML.

Another area that I will have to investigate further is the grammatical completeness of GCC-XML. GCC-XML supports many important C++ elements but some might have been omitted. This is an acknowledged shortcoming of GCC-XML. Some of the work to rectify this problem is started and mentioned on the GCC-XML mailing list but much of it has yet to be patched into the mainline of the code base. Once these modifications are made, I will have to make changes to the core Trike logic to accommodate them but I am not certain at this point where such changes might be made beyond modifications to the DDL statements, parsing of the XML, and un-

derlying SQL statements to insert the parsed data. The possibility also remains that GCC-XML might need to be replaced by a different tool like CIL.

5.1 Potential Limitations

Trike is currently limited by the completeness of GCC-XML and my current list of queries. Like many other Static Analysis tools Trike is limited in scope both by language and by the type of errors it can detect. It is conceivable that some of these limitations can easily be overcome by extensions such as small programs performing operations outside of the database, stored procedures and functions performing operations inside of the database, as well as improvements to the underlying components like GCC-XML which often forced me to use functions like **distinct**() to derive useful data from the stored XML.

5.2 Areas for Improvement

After the changes discussed in 5 are made I plan to make the interface to Trike even better. SQL is a useful tool for gathering data but not all developers have a complete grasp of the various standards and subtle nuances of particular implementations like those found in MariaDB. In order to make regular routines more useful and to enable the development of external code frameworks I am currently planning for Trike to include stored procedures and functions to make particular types of analysis programmatically accessible. Once this is accomplished I plan to create simple libraries for popular dynamic languages like Perl, PHP and Python to expose the power of Trike for users to include in their own work-flows.

Trike is a useful tool and most importantly an approachable tool. However, it is new and requires more work. The generated XML is potentially incomplete when compared to the various C++ specifications. The current DTD is useful but as the generated XML is further inspected, changes to the internals of GCC-XML will invariably result in modifications to the existing DTD. The database currently sup-

ports one project and unless the developer is exploring the *Partial Program Analysis* aspects of Trike, it is regenerated for every run. The final area that needs further exploration is the area of *Partial Program Analysis*. Trike is able to regenerate XML on a per file basis but how this regeneration cleanly fits within the entire database model requires further research before a concise replacement of specific elements can be guaranteed.

BIBLIOGRAPHY

- [1] Apple Computer. Xcode: Integrated development environment. <http://developer.apple.com/technologies/mac/#xcode>, July 2011.
- [2] A. Beaulieu. *Learning SQL*. O'Reilly, 1st edition, 2005.
- [3] B. Boehm. A view of 20th and 21st century software engineering. In *Proceedings of the 28th international conference on Software engineering*, ICSE '06, pages 12–29, New York, NY, USA, 2006. ACM.
- [4] B. W. Boehm. *Software Engineering Economics*. Prentice Hall, 2nd edition, 1981.
- [5] C. Cadar, D. Dunbar, and D. Engler. Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. Technical report, Stanford, Stanford University, 353 Serra Mall, Stanford, California 94305, 2008.
- [6] S. S. Chawathe. Describing and manipulating xml data. *Special Issue on XML, IEEE Computer Society*, 22(3):3–9, 1999.
- [7] H. Chen and D. Wagner. MOPS: an Infrastructure for Examining Security Properties of Software. In *Proceeding of the 9th ACM conference on Computer and communications security*, CCS '02, New York, NY, USA, 2002. ACM.
- [8] C. N. Christopher. Evaluating Static Analysis Frameworks. Technical report, Carnegie Mellon University, May 2006.
- [9] E. Codd. A relational model of data for large shared data banks. *Information Retrieval, Communications of the ACM*, 13(6):377–387, 1970.
- [10] Cppcheck Developers. Cppcheck. <http://cppcheck.sourceforge.net/>, June 2011.

- [11] A. Dasgupta, V. R. Narasayya, and M. Syamala. A static analysis framework for database applications. In *International Conference on Data Engineering*, pages 1403–1414, 2009.
- [12] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Maier, and D. Suciu. Querying xml data. *Special Issue on XML, IEEE Computer Society*, 22(3):10–18, 1999.
- [13] I. Dillig, T. Dillig, and A. Aiken. SAIL: Static Analysis Intermediate Language with a Two-Level Representation. Technical report, Stanford, 2009.
- [14] D. Engler and M. Musuvathi. Static Analysis versus Software Model Checking for Bug Finding. In *In VMCAI*, pages 191–210. Springer, 2004.
- [15] D. Evans and D. Larochelle. Improving security using extensible lightweight static analysis. *IEEE Softw.*, 19:42–51, January 2002.
- [16] N. Flemming, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer, 2nd edition, 2005.
- [17] D. Florescu and D. Kossmann. A performance evaluation of alternative mapping schemes for storing xml data in a relational database. Technical report, Inria, France and University of Passau, Germany, 1999.
- [18] D. Florescu and D. Kossmann. Storing and Querying XML Data Using an RDBMS. *Special Issue on XML, IEEE Computer Society*, 22(3):27–34, 1999.
- [19] A. Fokin, K. Troshina, and A. Chernov. Reconstruction of class hierarchies for decompilation of c++ programs. *Software Maintenance and Reengineering, European Conference on*, 0:240–243, 2010.
- [20] D. Galin. *Software Quality Assurance*. Pearson, 1st edition, 2004.
- [21] GCC-XML Developers. GCC-XML DTD. <http://www.gccxml.org/files/v0.6/gccxml-2004-11-19.dtd.txt>, June 2011.
- [22] GCC-XML Developers. GCC-XML Mailing List - DTD Beginnings. <http://www.cmake.org/pipermail/gccxml/2003-July/000237.html>, June 2011.

- [23] GCC-XML Developers. GCC-XML Mailing List - DTD Update. <http://www.cmake.org/pipermail/gccxml/2004-November/000511.html>, June 2011.
- [24] GCC-XML Developers. GCC-XML: The XML output extension to GCC. <http://www.gccxml.org/HTML/Index.html>, June 2011.
- [25] Google Inc. and CppLint Developers. CppLint. <http://sourceforge.net/projects/cpplint/>, June 2011.
- [26] T. Hofer. Evaluating Static Source Code Analysis Tools. Master's thesis, M-Icole Polytechnique FM-IdM-Irale de Lausanne, Lausanne, Switzerland, 2010.
- [27] D. Hovemeyer, J. Spacco, and W. Pugh. Evaluating and tuning a static analysis to find null pointer bugs. *SIGSOFT Softw. Eng. Notes*, 31:13–19, September 2005.
- [28] M. F. II, L. Marrs, and B. G. Ryder. Visualizing the Results of a Complex Hybrid Dynamic-Static Analysis. Technical report, Virginia Tech, 114 McBryde Hall (0106) Virginia Tech Blacksburg, VA 24061, 2010.
- [29] S. H. Kan. *Metrics and Models in Software Quality Engineering*. Pearson, 1st edition, 2003.
- [30] Y. P. Khoo, J. S. Foster, M. Hicks, and V. Sazawal. Path projection for user-centered static analysis tools. In *Proceedings of the 8th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering, PASTE '08*, pages 57–63, New York, NY, USA, 2008. ACM.
- [31] R. Krishnamurthy, R. Kaushik, and J. Naughton. XML-to-SQL Query Translation Literature: The State of the Art and Open Problems. In Z. Bellahsène, A. Chaudhri, E. Rahm, M. Rys, and R. Unland, editors, *Database and XML Technologies*, volume 2824 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin / Heidelberg, 2003. 10.1007/978-3-540-39429-7_1.

- [32] B. Livshits. *Improving Software Security with Precise Static and Runtime Analysis*. PhD thesis, Stanford University, Palo Alto, CA, USA, 2006.
- [33] LLVM Developers. The Clang Static Analyzer. <http://clang-analyzer.llvm.org/>, June 2011.
- [34] M. O. McCracken. The Design and Implementation of the LENS Program Information Framework. Technical report, UCSD, 2006.
- [35] J. Midtgaard. Control-flow Analysis of Functional Programs. Technical report, University of Aarhus, IRISA/INRIA, BRICS, Department of Computer Science, University of Aarhus, IT-parken, Aabogade 34, DK-8200, Aarhus, N, Denmark, November 2007.
- [36] A. Mockus and D. M. Weiss. Predicting risk of software changes. *Bell Labs Technical Journal*, 5:169–180, 2000.
- [37] M. S. Musuvathi, D. Park, D. Y. W. Park, A. Chou, D. R. Engler, and D. L. Dill. Cmc: A pragmatic approach to model checking real code. In *In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, 2002.
- [38] N. Nagappan and T. Ball. Static analysis tools as early indicators of pre-release defect density. In *Proceedings of the 27th international conference on Software engineering*, ICSE '05, pages 580–586, New York, NY, USA, 2005. ACM.
- [39] N. Nagappan, T. Ball, and A. Zeller. Mining Metrics to Predict Component Failures. Technical report, Microsoft Research, 2005.
- [40] G. C. Necula, S. McPeak, S. P. Rahul, and W. Weimer. CIL: Intermediate Language and Tools for Analysis and Transformation of C Programs. In *In International Conference on Compiler Construction*, pages 213–228, 2002.
- [41] Oracle, Inc. MySQL. <http://dev.mysql.com>, June 2011.

- [42] D. Quinlan and C. Cohen. Discovery of C++ Data Structures from Binaries. *Cyber Security and Information Intelligence Research Workshop, ACM 6th Annual*, 0:1–3, 2010.
- [43] N. Rutar, C. B. Almazan, and J. S. Foster. A Comparison of Bug Finding Tools for Java. In *Proceedings of the 15th International Symposium on Software Reliability Engineering*, pages 245–256, Washington, DC, USA, 2004. IEEE Computer Society.
- [44] SchemaSpy Developers. SchemaSpy: Graphical Database Schema Metadata Browser. <http://schemaspy.sourceforge.net/>, July 2011.
- [45] B. Schwarz, H. Chen, D. Wagner, J. Lin, W. Tu, G. Morrison, and J. West. Model checking an entire linux distribution for security violations. In *Proceedings of the 21st Annual Computer Security Applications Conference*, pages 13–22, Washington, DC, USA, 2005. IEEE Computer Society.
- [46] R. W. Selby. *Software Engineering: Barry W. Boehm’s Lifetime Contributions to Software Development, Management, and Research*. IEEE Computer Society, 1st edition, 2007.
- [47] I. Tatarinov, S. D. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang. Storing and querying ordered xml using a relational database system. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, SIGMOD ’02, pages 204–215, New York, NY, USA, 2002. ACM.
- [48] The Eclipse Foundation. Eclipse: Integrated Development Environemnt. <http://www.eclipse.org/>, July 2011.
- [49] The GNU Project. GCC: The GNU Compiler Collection. <http://gcc.gnu.org>, July 2011.
- [50] K. K. —ver Jensen. Type Checking of SQL/XML. Master’s thesis, University of Aarhus, Denmark, 2010.

- [51] W3C: World-Wide Web Consortium. Extensible Markup Language (XML). <http://www.w3.org/XML/>, July 2011.
- [52] W3C: World-Wide Web Consortium. W3C XML Query (XQuery). <http://www.w3.org/XML/Query/>, July 2011.
- [53] W3C: World-Wide Web Consortium. XML Path Language (XPath). <http://www.w3.org/TR/xpath/>, July 2011.
- [54] W3C: World-Wide Web Consortium. XSL Transformations (XSLT) Version 2.0. <http://www.w3.org/TR/xslt20/>, July 2011.
- [55] M. Widenius. MariaDB. <http://mariadb.org>, June 2011.
- [56] D. Wilkerson, K. Chen, and S. McPeak. Oink: a Collaboration of C/C++ Tools for Static Analysis and Source-to-Source Transformation. <http://daniel-wilkerson.appspot.com/oink/index.html>, June 2011.
- [57] T. Witkowski, N. Blanc, D. Kroening, and G. Weissenbacher. Model checking concurrent Linux device drivers. In *22nd IEEE International Conference on Automated Software Engineering (ASE)*, pages 501–504. IEEE, 2007.
- [58] D. Worth, C. Greenough, and L. Chin. A Survey of C and C++ Software Tools for Computational Science. Technical report, Science and Technology Facilities Council, Library and Information Services, SFTC Rutherford Appleton Laboratory, Harwell Science and Innovation Campus, Didcot OX11 0QX, UK, December 2009.
- [59] V. Yerramilli. Static Analysis of Novice Student C++ Programs. Master’s thesis, Texas Tech University, Texas, USA, 1998.
- [60] A. Zeller. Learning from 6,000 projects: Mining models in the large. *Source Code Analysis and Manipulation, IEEE International Workshop on*, 0:3–6, 2010.

- [61] H. Zheng, K. Zhou, X. Lai, C. Liu, and Z. Chi. Research on XML Based Static Analysis Software Security Analysis. In *Software Engineering (WCSE), 2010 Second World Congress On*, volume 2, pages 141–144, 2010.

APPENDIX

A-1 Structural Query Output

A-1.1 The Number of Files

The output for the libodbc++ project for this query is

The Number of Files in libodbc++

```
1 +-----+
2 | Number of Files |
3 +-----+
4 |           42 |
5 +-----+
6 1 row in set (0.00 sec)
```

for the log4cxx project the output is

The Number of Files in log4cxx

```
1 +-----+
2 | Number of Files |
3 +-----+
4 |           325 |
5 +-----+
6 1 row in set (0.00 sec)
```

and for the C++ Sockets Library the output is

The Number of Files in the C++ Sockets Library

```
1 +-----+
2 | Number of Files |
3 +-----+
4 |           122 |
5 +-----+
6 1 row in set (0.00 sec)
```

A-1.2 The Number of Header Files

The output for the libodbc++ project for this query is

The Number of Header Files in libodbc++

```
1 +-----+
2 | Number of Header Files |
3 +-----+
4 |                23 |
5 +-----+
6 1 row in set (0.00 sec)
```

for the log4cxx project the output is

The Number of Header Files in log4cxx

```
1 +-----+
2 | Number of Header Files |
3 +-----+
4 |                177 |
5 +-----+
6 1 row in set (0.00 sec)
```

and for the C++ Sockets Library the output is

The Number of Header Files in the C++ Sockets Library

```
1 +-----+
2 | Number of Header Files |
3 +-----+
4 |                67 |
5 +-----+
6 1 row in set (0.00 sec)
```

A-1.3 The Number of Source Files

The output for the libodbc++ project for this query is

The Number of Source Files in libodbc++

```
1 +-----+
2 | Number of Source Files |
3 +-----+
4 |                19 |
5 +-----+
6 1 row in set (0.00 sec)
```

for the log4cxx project the output is

The Number of Source Files in log4cxx

```
1 +-----+
2 | Number of Source Files |
3 +-----+
4 |                148 |
5 +-----+
6 1 row in set (0.00 sec)
```

and for the C++ Sockets Library the output is

The Number of Source Files in the C++ Sockets Library

```
1 +-----+
2 | Number of Source Files |
3 +-----+
4 |                55 |
5 +-----+
6 1 row in set (0.00 sec)
```

A-1.4 Total Source Lines of Code (SLOC)

The output for the libodbc++ project for this query is

Total SLOC for libodbc++

```
1 +-----+
2 | SLOC |
3 +-----+
4 | 16444 |
5 +-----+
6 1 row in set (0.11 sec)
```

for the log4cxx project the output is

Total SLOC for log4cxx

```
1 +-----+
2 | SLOC |
3 +-----+
4 | 45733 |
5 +-----+
6 1 row in set (0.20 sec)
```

and for the C++ Sockets Library the output is

Total SLOC for the C++ Sockets Library

```
1 +-----+
2 | SLOC |
3 +-----+
4 | 26869 |
5 +-----+
6 1 row in set (0.13 sec)
```

A-1.5 The SLOC per File

The output for the libodbc++ project for this query is

The SLOC for each file in libodbc++

```
1 +-----+-----+-----+
2 | id | name                | SLOC |
3 +-----+-----+-----+
4 | 1 | statement.cpp        | 786 |
5 | 2 | driverinfo.h         | 145 |
6 | 3 | resultsetmetadata.cpp | 366 |
7 | 4 | threads.cpp          | 123 |
8 | 5 | errorhandler.cpp     | 302 |
9 | 6 | drivermanager.cpp    | 369 |
10 | 7 | datastream.cpp       | 293 |
11 | 8 | callablestatement.cpp | 96 |
12 | 9 | dtconv.h             | 324 |
13 | 10 | datetime.cpp         | 316 |
14 | 11 | datahandler.h        | 279 |
15 | 12 | connection.cpp       | 504 |
16 | 13 | preparedstatement.cpp | 486 |
17 | 14 | datahandler.cpp      | 1000 |
18 | 15 | databasemetadata.cpp | 1766 |
19 | 16 | datastream.h         | 206 |
20 | 17 | driverinfo.cpp       | 135 |
```

```

21 | 18 | resultset.cpp          | 1552 |
22 | 19 | types.h               | 977  |
23 | 20 | resultsetmetadata.h  | 193  |
24 | 21 | config-win32.h       | 106  |
25 | 22 | statement.h          | 270  |
26 | 23 | drivermanager.h     | 164  |
27 | 24 | databasemetadata.h  | 1349 |
28 | 25 | threads.h            | 79   |
29 | 26 | callablestatement.h | 166  |
30 | 27 | errorhandler.h       | 141  |
31 | 28 | config.h             | 173  |
32 | 28 | config.h             | 170  |
33 | 29 | connection.h         | 238  |
34 | 30 | preparedstatement.h | 229  |
35 | 31 | resultset.h          | 629  |
36 | 32 | setup.h              | 220  |
37 | 33 | mainwindow.cpp       | 257  |
38 | 34 | resultwindow.h       | 83   |
39 | 35 | connectwindow.cpp    | 115  |
40 | 36 | connectwindow.h      | 60   |
41 | 37 | resultwindow.cpp     | 214  |
42 | 38 | main.cpp             | 35   |
43 | 39 | mainwindow.h         | 71   |
44 | 40 | isql++.cpp           | 1321 |
45 | 41 | isql++.h             | 136  |
46 +-----+-----+-----+-----+
47 42 rows in set (0.11 sec)

```

for the log4cxx project the output is

The SLOC for each file in log4cxx

```

1 +-----+-----+-----+-----+
2 | id  | name                               | SLOC |
3 +-----+-----+-----+-----+
4 | 1   | defaultconfigurator.h             | 55   |
5 | 2   | odbccappender.h                   | 292  |
6 | 3   | rollingfileappender.h             | 116  |
7 | 3   | rollingfileappender.h             | 105  |
8 | 4   | ndc.h                              | 353  |
9 | 5   | layout.h                          | 97   |
10 | 6   | hierarchy.h                       | 283  |
11 | 7   | htmllayout.h                      | 133  |
12 | 8   | basicconfigurator.h               | 65   |
13 | 9   | file.h                            | 188  |
14 | 10  | provisionnode.h                   | 35   |

```

15		11		xmllayout.h		138	
16		12		domconfigurator.h		316	
17		13		levelrangefilter.h		137	
18		14		andfilter.h		109	
19		15		mapfilter.h		44	
20		16		expressionfilter.h		130	
21		17		stringmatchfilter.h		92	
22		18		levelmatchfilter.h		102	
23		19		propertyfilter.h		81	
24		20		denyallfilter.h		73	
25		21		locationinfofilter.h		87	
26		22		loggerfactory.h		46	
27		23		rootlogger.h		60	
28		24		hierarchyeventlistener.h		65	
29		25		defaultrepositoryselector.h		50	
30		26		optionhandler.h		72	
31		27		triggeringeventevaluator.h		50	
32		28		repositoryselector.h		54	
33		29		appenderattachable.h		98	
34		30		locationinfo.h		136	
35		31		errorhandler.h		127	
36		32		filter.h		133	
37		33		loggerrepository.h		116	
38		34		configurator.h		61	
39		35		loggingevent.h		263	
40		36		defaultloggerfactory.h		45	
41		37		filewatchdog.h		86	
42		38		iso8601dateformat.h		46	
43		39		outputstream.h		59	
44		40		stringhelper.h		61	
45		41		dateformat.h		92	
46		42		datelayout.h		91	
47		43		inetaddress.h		106	
48		44		optionconverter.h		164	
49		45		bytearrayoutputstream.h		77	
50		46		appenderattachableimpl.h		128	
51		47		strictmath.h		49	
52		48		cacheddateformat.h		220	
53		49		transform.h		62	
54		50		synchronized.h		44	
55		51		class.h		67	
56		52		locale.h		49	
57		53		objectptr.h		183	
58		54		xml.h		131	
59		55		resourcebundle.h		89	

60		56		reader.h		72	
61		57		aprinitializer.h		57	
62		58		socketoutputstream.h		74	
63		59		strftimedateformat.h		68	
64		60		condition.h		73	
65		61		datagramsocket.h		128	
66		62		simpledateformat.h		96	
67		63		bufferedoutputstream.h		63	
68		64		timezone.h		70	
69		65		absolutetimedateformat.h		40	
70		66		classregistration.h		41	
71		67		properties.h		186	
72		68		bytearrayinputstream.h		94	
73		69		bufferedwriter.h		63	
74		70		onlyonceerrorhandler.h		103	
75		71		propertyresourcebundle.h		62	
76		72		date.h		67	
77		73		system.h		53	
78		74		mutex.h		51	
79		75		stringtokenizer.h		49	
80		76		serversocket.h		64	
81		77		outputstreamwriter.h		65	
82		78		threadspecificdata.h		67	
83		79		fileoutputstream.h		67	
84		80		inputstream.h		72	
85		81		socket.h		90	
86		82		charsetdecoder.h		113	
87		83		thread.h		184	
88		84		systemerrwriter.h		59	
89		85		objectimpl.h		49	
90		86		integer.h		50	
91		87		loglog.h		113	
92		88		object.h		138	
93		89		fileinputstream.h		96	
94		90		transcoder.h		256	
95		91		datagrampacket.h		135	
96		92		cyclicbuffer.h		92	
97		93		relativetimedateformat.h		47	
98		94		datetimedateformat.h		42	
99		95		syslogwriter.h		48	
100		96		tchar.h		171	
101		97		inputstreamreader.h		94	
102		98		messagebuffer.h		828	
103		99		loader.h		43	
104		100		charsetencoder.h		135	

105		101		bytebuffer.h		69	
106		102		systemoutwriter.h		58	
107		103		exception.h		287	
108		104		objectoutputstream.h		95	
109		105		pool.h		59	
110		106		threadlocal.h		85	
111		107		writer.h		58	
112		108		log4cxx.h		51	
113		109		rollingpolicy.h		82	
114		111		rolloverdescription.h		101	
115		112		filterbasedtriggeringpolicy.h		118	
116		113		rollingpolicybase.h		128	
117		114		action.h		87	
118		115		timebasedrollingpolicy.h		227	
119		116		triggeringpolicy.h		80	
120		117		rollingfileappenderskeleton.h		152	
121		118		manualtriggeringpolicy.h		72	
122		119		filerenameaction.h		59	
123		120		sizebasedtriggeringpolicy.h		82	
124		121		gzcompressaction.h		73	
125		122		zipcompressaction.h		74	
126		123		fixedwindowrollingpolicy.h		141	
127		124		log4cxx_private.h		56	
128		125		methodlocationpatternconverter.h		61	
129		126		patternparser.h		171	
130		127		ndcpatternconverter.h		60	
131		128		filedatepatternconverter.h		52	
132		129		loggerpatternconverter.h		67	
133		130		formattinginfo.h		117	
134		131		fulllocationpatternconverter.h		63	
135		132		loggingeventpatternconverter.h		84	
136		133		threadpatternconverter.h		60	
137		134		linelocationpatternconverter.h		62	
138		135		levelpatternconverter.h		62	
139		136		throwableinformationpatternconverter.h		74	
140		137		lineseparatorpatternconverter.h		69	
141		138		datepatternconverter.h		82	
142		139		literalpatternconverter.h		67	
143		140		namepatternconverter.h		74	
144		141		nameabbreviator.h		81	
145		142		classnamepatternconverter.h		65	
146		143		filelocationpatternconverter.h		62	
147		144		relativetimepatternconverter.h		59	
148		145		patternconverter.h		124	
149		146		propertiespatternconverter.h		71	

150		147		integerpatternconverter.h		64	
151		148		messagepatternconverter.h		60	
152		149		asynccappender.h		292	
153		150		portability.h		25	
154		151		dailyrollingfileappender.h		203	
155		152		logmanager.h		212	
156		153		appender.h		148	
157		154		fileappender.h		223	
158		155		consoleappender.h		79	
159		156		propertysetter.h		108	
160		157		nteventlogappender.h		111	
161		158		outputdebugstringappender.h		48	
162		159		writerappender.h		217	
163		160		socketappender.h		143	
164		161		syslogappender.h		141	
165		162		telnetappender.h		158	
166		163		socketappenderskeleton.h		179	
167		164		sockethubappender.h		197	
168		165		smtpappender.h		283	
169		166		xmlsocketappender.h		150	
170		167		logstring.h		90	
171		168		simplelayout.h		83	
172		169		level.h		285	
173		170		fallbackerrorhandler.h		116	
174		171		appenderskeleton.h		213	
175		172		mdc.h		240	
176		173		ttcclayout.h		187	
177		174		propertyconfigurator.h		395	
178		175		stream.h		566	
179		176		logger.h		1911	
180		177		patternlayout.h		409	
181		178		integer.cpp		34	
182		179		locationinfo.cpp		195	
183		180		properties.cpp		370	
184		181		file.cpp		240	
185		182		inputstreamreader.cpp		74	
186		183		hierarchy.cpp		399	
187		184		socket.cpp		130	
188		185		socketappender.cpp		111	
189		186		aprinitializer.cpp		69	
190		187		propertiespatternconverter.cpp		80	
191		188		ndc.cpp		338	
192		189		threadpatternconverter.cpp		50	
193		190		basicconfigurator.cpp		45	
194		191		fulllocationpatternconverter.cpp		56	

195		192		propertyconfigurator.cpp		451	
196		193		nameabbreviator.cpp		325	
197		194		objectoutputstream.cpp		194	
198		195		outputstream.cpp		30	
199		196		socketappenderskeleton.cpp		176	
200		197		logstream.cpp		504	
201		198		resourcebundle.cpp		122	
202		199		relativetimepatternconverter.cpp		53	
203		200		charsetencoder.cpp		518	
204		201		inetaddress.cpp		144	
205		202		throwableinformationpatternconverter.cpp		64	
206		203		patternconverter.cpp		50	
207		204		exception.cpp		427	
208		205		rollingpolicybase.cpp		146	
209		206		classregistration.cpp		28	
210		207		stringmatchfilter.cpp		80	
211		208		datagrampacket.cpp		58	
212		209		datagramsocket.cpp		186	
213		210		layout.cpp		40	
214		211		threadlocal.cpp		60	
215		212		levelmatchfilter.cpp		84	
216		213		htmlayout.cpp		220	
217		214		fallbackerrorhandler.cpp		106	
218		215		writer.cpp		29	
219		216		charsetdecoder.cpp		490	
220		217		obsoleterollingfileappender.cpp		155	
221		218		triggeringpolicy.cpp		35	
222		219		fileoutputstream.cpp		98	
223		220		patternparser.cpp		341	
224		221		integerpatternconverter.cpp		52	
225		222		systemerrwriter.cpp		73	
226		223		transform.cpp		117	
227		224		sizebasedtriggeringpolicy.cpp		56	
228		225		onlyonceerrorhandler.cpp		89	
229		226		outputstreamwriter.cpp		78	
230		227		writerappender.cpp		264	
231		228		datepatternconverter.cpp		135	
232		229		class.cpp		192	
233		230		telnetappender.cpp		252	
234		231		rolloverdescription.cpp		62	
235		232		nventlogappender.cpp		316	
236		233		dateformat.cpp		37	
237		234		filterbasedtriggeringpolicy.cpp		95	
238		235		objectptr.cpp		47	
239		236		timezone.cpp		250	

240		237		systemoutwriter.cpp		74	
241		238		patternlayout.cpp		187	
242		239		asynccappender.cpp		364	
243		240		objectimpl.cpp		48	
244		241		filedatepatternconverter.cpp		39	
245		242		sockethubappender.cpp		238	
246		243		filerenameaction.cpp		35	
247		244		fileinputstream.cpp		94	
248		245		zipcompressaction.cpp		86	
249		246		system.cpp		96	
250		247		logmanager.cpp		211	
251		248		fixedwindowrollingpolicy.cpp		295	
252		249		stringhelper.cpp		159	
253		250		filewatchdog.cpp		93	
254		251		cacheddateformat.cpp		309	
255		252		appenderattachableimpl.cpp		160	
256		253		reader.cpp		29	
257		254		formattinginfo.cpp		68	
258		255		odbcappender.cpp		333	
259		256		methodlocationpatternconverter.cpp		51	
260		257		bufferedwriter.cpp		60	
261		258		bytearrayinputstream.cpp		55	
262		259		strftimedateformat.cpp		55	
263		260		syslogappender.cpp		356	
264		261		rollingpolicy.cpp		24	
265		262		rollingfileappender.cpp		384	
266		263		loggingevent.cpp		326	
267		264		gzcompressaction.cpp		95	
268		265		namepatternconverter.cpp		56	
269		266		outputdebugstringappender.cpp		47	
270		267		loggingeventpatternconverter.cpp		49	
271		268		simpledateformat.cpp		781	
272		269		filelocationpatternconverter.cpp		50	
273		270		threadspecificdata.cpp		131	
274		271		xmlsocketappender.cpp		118	
275		272		inputstream.cpp		30	
276		273		optionconverter.cpp		381	
277		274		appenderskeleton.cpp		177	
278		275		date.cpp		53	
279		276		locale.cpp		55	
280		277		propertysetter.cpp		104	
281		278		synchronized.cpp		49	
282		279		propertyresourcebundle.cpp		57	
283		280		bytebuffer.cpp		65	
284		281		literalpatternconverter.cpp		62	

285		282		simplelayout.cpp		37		
286		283		loggerpatternconverter.cpp		55		
287		284		levelrangefilter.cpp		89		
288		285		configurator.cpp		32		
289		286		ttcclayout.cpp		77		
290		287		syslogwriter.cpp		68		
291		288		manualtriggeringpolicy.cpp		42		
292		289		messagebuffer.cpp		430		
293		290		level.cpp		234		
294		291		threadcxx.cpp		175		
295		292		cyclicbuffer.cpp		148		
296		293		loglog.cpp		121		
297		294		fileappender.cpp		309		
298		295		mutex.cpp		61		
299		296		stringtokenizer.cpp		60		
300		297		filter.cpp		49		
301		298		datelayout.cpp		121		
302		299		action.cpp		75		
303		300		defaultloggerfactory.cpp		30		
304		301		loader.cpp		71		
305		302		domconfigurator.cpp		969		
306		303		levelpatternconverter.cpp		87		
307		304		relativetimedateformat.cpp		38		
308		305		serversocket.cpp		140		
309		306		timebasedrollingpolicy.cpp		193		
310		307		defaultconfigurator.cpp		105		
311		308		smtpappender.cpp		674		
312		309		rootlogger.cpp		52		
313		310		pool.cpp		86		
314		311		socketoutputstream.cpp		57		
315		312		messagepatternconverter.cpp		52		
316		313		transcoder.cpp		538		
317		314		dailyrollingfileappender.cpp		108		
318		315		defaultrepositoryselector.cpp		40		
319		316		condition.cpp		70		
320		317		lineseparatorpatternconverter.cpp		57		
321		318		classnamepatternconverter.cpp		56		
322		319		logger.cpp		975		
323		320		mdc.cpp		236		
324		321		xmllayout.cpp		146		
325		322		ndcpatternconverter.cpp		54		
326		323		bytearrayoutputstream.cpp		53		
327		324		consoleappender.cpp		146		
328		325		linelocationpatternconverter.cpp		52		
329		+-----+-----+-----+-----+-----+						

and for the C++ Sockets Library the output is

The SLOC for each file in the C++ Sockets Library

id	name	SLOC
1	Event.h	80
2	Debug.cpp	103
3	IHttpServer.h	68
4	SocketStream.cpp	69
5	HttpRequest.h	140
6	EventTime.h	79
7	HttpRequest.cpp	428
8	HttpPostSocket.h	98
9	Thread.cpp	178
10	HttpBaseSocket.h	87
11	MemFile.cpp	349
12	Ajp13Socket.cpp	414
13	HttpPostSocket.cpp	295
14	XmlException.cpp	52
15	SocketThread.cpp	51
16	EventHandler.h	85
17	Exception.h	71
18	IEventOwner.h	79
19	Lock.h	61
20	Sockets-config.cpp	121
21	Ipv4Address.h	108
22	ResolveServer.cpp	99
23	StreamSocket.cpp	173
24	SctpSocket.h	119
25	HttpResponse.cpp	251
26	EventHandler.cpp	209
27	SocketStream.h	66
28	SocketHandlerEp.cpp	216
29	UdpSocket.cpp	905
30	Mutex.cpp	83
31	UdpSocket.h	224
32	StdoutLog.cpp	108
33	ResolveSocket.cpp	442
34	StreamWriter.h	67
35	HttpdCookies.cpp	286
36	FileStream.h	61
37	File.cpp	203

41		38		Utility.cpp		1435	
42		39		XmlDocument.h		79	
43		40		sockets-config.h		110	
44		41		EventTime.cpp		107	
45		42		HTTPSocket.cpp		518	
46		43		SctpSocket.cpp		506	
47		44		SocketThread.h		40	
48		45		socket_include.cpp		93	
49		46		Parse.cpp		323	
50		47		IFileUpload.h		60	
51		48		AjpBaseSocket.h		95	
52		49		Debug.h		78	
53		50		HttpdForm.h		141	
54		51		SocketHandlerThread.h		67	
55		52		SSLInitializer.cpp		181	
56		53		Ipv6Address.h		117	
57		54		SocketHandlerThread.cpp		81	
58		55		SmtpdSocket.cpp		244	
59		56		XmlNode.h		150	
60		57		Thread.h		128	
61		58		Base64.cpp		314	
62		59		SocketHandler.h		295	
63		60		Lock.cpp		57	
64		61		XmlException.h		65	
65		62		HttpTransaction.cpp		262	
66		63		FileStream.cpp		60	
67		64		Base64.h		82	
68		65		TcpSocket.cpp		1850	
69		66		StreamWriter.cpp		92	
70		67		SocketHandler.cpp		1418	
71		68		Json.h		96	
72		69		Mutex.h		72	
73		70		Exception.cpp		66	
74		71		IStream.h		60	
75		72		HttpBaseSocket.cpp		215	
76		73		HTTPSocket.h		168	
77		74		IFile.h		74	
78		75		ListenSocket.h		512	
79		76		ResolvSocket.h		109	
80		77		HttpResponse.h		98	
81		78		Semaphore.h		97	
82		79		IEventOwner.cpp		81	
83		80		IBase.h		44	
84		81		TcpSocket.h		365	
85		82		Semaphore.cpp		123	

86		83		XmlDocument.cpp		102		
87		84		HttpDebugSocket.h		73		
88		85		AjpBaseSocket.cpp		268		
89		86		SSLInitializer.h		96		
90		87		HttpDebugSocket.cpp		144		
91		88		socket_include.h		298		
92		89		SocketAddress.h		105		
93		90		StdLog.h		76		
94		91		HttpdSocket.h		108		
95		92		HttpPutSocket.h		85		
96		93		ISocketHandler.h		261		
97		94		HttpdCookies.h		91		
98		95		ResolvServer.h		75		
99		96		Ipv6Address.cpp		285		
100		97		XmlNode.cpp		357		
101		98		MemFile.h		115		
102		99		SmtpdSocket.h		153		
103		100		HttpGetSocket.h		68		
104		101		HttpClientSocket.h		134		
105		102		Ajp13Socket.h		80		
106		103		Event.cpp		93		
107		104		HttpPutSocket.cpp		125		
108		105		HttpdSocket.cpp		326		
109		106		HttpTransaction.h		105		
110		107		Ipv4Address.cpp		223		
111		108		IEventHandler.h		77		
112		109		Socket.h		720		
113		110		HttpClientSocket.cpp		312		
114		111		IMutex.h		57		
115		112		Parse.h		103		
116		113		Json.cpp		558		
117		114		Socket.cpp		1875		
118		115		StreamSocket.h		139		
119		116		StdoutLog.h		63		
120		117		HttpdForm.cpp		716		
121		118		File.h		92		
122		119		ajp13.h		101		
123		120		Utility.h		245		
124		121		SocketHandlerEp.h		95		
125		122		HttpGetSocket.cpp		114		
126		+-----+-----+-----+-----+						
127		122 rows in set (0.14 sec)						

A-1.6 Total Header SLOC

The output for the libodbc++ project for this query is

Total Header SLOC for libodbc++

```
1 +-----+
2 | The SLOC in Header Files |
3 +-----+
4 |                6408 |
5 +-----+
6 1 row in set (0.04 sec)
```

for the log4cxx project the output is

Total Header SLOC for log4cxx

```
1 +-----+
2 | The SLOC in Header Files |
3 +-----+
4 |                22356 |
5 +-----+
6 1 row in set (0.07 sec)
```

and for the C++ Sockets Library the output is

Total Header SLOC for the C++ Sockets Library

```
1 +-----+
2 | The SLOC in Header Files |
3 +-----+
4 |                8310 |
5 +-----+
6 1 row in set (0.04 sec)
```

A-1.7 Total Source SLOC

The output for the libodbc++ project for this query is

Total Source SLOC for libodbc++

```
1 +-----+
2 | The SLOC in Source Files |
3 +-----+
4 |                10036 |
5 +-----+
6 1 row in set (0.07 sec)
```

for the log4cxx project the output is

Total Source SLOC for log4cxx

```
1 +-----+
2 | The SLOC in Source Files |
3 +-----+
4 |                23377 |
5 +-----+
6 1 row in set (0.04 sec)
```

and for the C++ Sockets Library the output is

Total Source SLOC for the C++ Sockets Library

```
1 +-----+
2 | The SLOC in Header Files |
3 +-----+
4 |                18559 |
5 +-----+
6 1 row in set (0.13 sec)
```

A-1.8 The Bytes per File

The output for the libodbc++ project for this query is

The Bytes for each File in libodbc++

```
1 +-----+-----+
2 | name                | Byte Count |
3 +-----+-----+
4 | statement.cpp       |    22497 |
5 | driverinfo.h        |     3819 |
6 | resultsetmetadata.cpp |     9413 |
7 | threads.cpp         |     2539 |
8 | errorhandler.cpp    |     7824 |
9 | drivermanager.cpp   |     9233 |
```

```

10 | datastream.cpp          |          7084 |
11 | callablestatement.cpp  |          2891 |
12 | dtconv.h               |          8183 |
13 | datetime.cpp           |          7173 |
14 | datahandler.h          |          6625 |
15 | connection.cpp         |         13910 |
16 | preparedstatement.cpp  |         12716 |
17 | datahandler.cpp        |         24347 |
18 | databasemetadata.cpp  |         43360 |
19 | datastream.h           |          4999 |
20 | driverinfo.cpp         |          4941 |
21 | resultset.cpp          |         39669 |
22 | types.h                |         23812 |
23 | resultsetmetadata.h    |          5724 |
24 | config-win32.h         |          2154 |
25 | statement.h            |          8965 |
26 | drivermanager.h        |          4777 |
27 | databasemetadata.h    |         49091 |
28 | threads.h              |          1760 |
29 | callablestatement.h    |          5052 |
30 | errorhandler.h         |          3964 |
31 | config.h               |          5111 |
32 | config.h               |          4607 |
33 | connection.h           |          7484 |
34 | preparedstatement.h    |          6936 |
35 | resultset.h            |         18732 |
36 | setup.h                |          6032 |
37 | mainwindow.cpp         |          5895 |
38 | resultwindow.h         |          1797 |
39 | connectwindow.cpp      |          2911 |
40 | connectwindow.h        |          1479 |
41 | resultwindow.cpp       |          4405 |
42 | main.cpp               |          1115 |
43 | mainwindow.h           |          1698 |
44 | isql++.cpp             |         30837 |
45 | isql++.h               |          2569 |
46 +-----+-----+
47 42 rows in set (0.00 sec)

```

for the log4cxx project the output is

The Bytes for each File in log4cxx

```

1 +-----+-----+
2 | name                               | Byte Count |
3 +-----+-----+

```

4	defaultconfigurator.h		1747	
5	odbcapender.h		12177	
6	rollingfileappender.h		4474	
7	rollingfileappender.h		4069	
8	ndc.h		15652	
9	layout.h		3444	
10	hierarchy.h		10162	
11	htmllayout.h		4749	
12	basicconfigurator.h		2202	
13	file.h		7377	
14	provisionnode.h		1168	
15	xmllayout.h		5638	
16	domconfigurator.h		13966	
17	levelrangefilter.h		6075	
18	andfilter.h		3710	
19	mapfilter.h		1356	
20	expressionfilter.h		4363	
21	stringmatchfilter.h		4076	
22	levelmatchfilter.h		4153	
23	propertyfilter.h		2506	
24	denyallfilter.h		2768	
25	locationinfofilter.h		3087	
26	loggerfactory.h		1663	
27	rootlogger.h		2213	
28	hierarchyeventlistener.h		2243	
29	defaultrepositoryselector.h		2028	
30	optionhandler.h		2966	
31	triggeringeventevaluator.h		2029	
32	repositoryselector.h		2160	
33	appenderattachable.h		2945	
34	locationinfo.h		3700	
35	errorhandler.h		5154	
36	filter.h		5201	
37	loggerrepository.h		4456	
38	configurator.h		1905	
39	loggingevent.h		11381	
40	defaultloggerfactory.h		1698	
41	filewatchdog.h		3105	
42	iso8601dateformat.h		1790	
43	outputstream.h		1972	
44	stringhelper.h		2309	
45	dateformat.h		3416	
46	datelayout.h		3762	
47	inetaddress.h		3775	
48	optionconverter.h		7221	

49		bytearrayoutputstream.h		2440	
50		appenderattachableimpl.h		4101	
51		strictmath.h		1837	
52		cacheddateformat.h		8194	
53		transform.h		2829	
54		synchronized.h		1542	
55		class.h		2199	
56		locale.h		1927	
57		objectptr.h		5610	
58		xml.h		4916	
59		resourcebundle.h		3648	
60		reader.h		2275	
61		aprintializer.h		1754	
62		socketoutputstream.h		2589	
63		strftimedateformat.h		2236	
64		condition.h		2874	
65		datagramsocket.h		5290	
66		simplifiedateformat.h		3132	
67		bufferedoutputstream.h		2198	
68		timezone.h		2064	
69		absolutetimedateformat.h		1545	
70		classregistration.h		1539	
71		properties.h		9067	
72		bytearrayinputstream.h		3104	
73		bufferedwriter.h		2175	
74		onlyonceerrorhandler.h		3895	
75		propertyresourcebundle.h		2534	
76		date.h		1790	
77		system.h		1838	
78		mutex.h		1629	
79		stringtokenizer.h		1871	
80		serversocket.h		2278	
81		outputstreamwriter.h		2358	
82		threadspecificdata.h		2547	
83		fileoutputstream.h		2404	
84		inputstream.h		2485	
85		socket.h		3477	
86		charsetdecoder.h		4033	
87		thread.h		7856	
88		systemerrwriter.h		2069	
89		objectimpl.h		1542	
90		integer.h		1387	
91		loglog.h		4483	
92		object.h		5200	
93		fileinputstream.h		3475	

94		transcoder.h		8350	
95		datagrampacket.h		6167	
96		cyclicbuffer.h		3873	
97		relativetimedateformat.h		1814	
98		datetimedateformat.h		1725	
99		syslogwriter.h		1718	
100		tchar.h		7344	
101		inputstreamreader.h		3406	
102		messagebuffer.h		26003	
103		loader.h		1534	
104		charsetencoder.h		4585	
105		bytebuffer.h		2243	
106		systemoutwriter.h		2094	
107		exception.h		13439	
108		objectoutputstream.h		3647	
109		pool.h		2012	
110		threadlocal.h		2887	
111		writer.h		1871	
112		log4cxx.h		1560	
113		rollingpolicy.h		3045	
114		rolloverdescription.h		3476	
115		filterbasedtriggeringpolicy.h		3195	
116		rollingpolicybase.h		3927	
117		action.h		2285	
118		timebasedrollingpolicy.h		9684	
119		triggeringpolicy.h		2794	
120		rollingfileappenderskeleton.h		5120	
121		manualtriggeringpolicy.h		2590	
122		filerenameaction.h		1814	
123		sizebasedtriggeringpolicy.h		2784	
124		gzcompressaction.h		2099	
125		zipcompressaction.h		2119	
126		fixedwindowrollingpolicy.h		5051	
127		log4cxx_private.h		1770	
128		methodlocationpatternconverter.h		1906	
129		patternparser.h		5421	
130		ndcpatternconverter.h		1792	
131		filedatepatternconverter.h		1566	
132		loggerpatternconverter.h		1934	
133		formattinginfo.h		2665	
134		fulllocationpatternconverter.h		1863	
135		loggingeventpatternconverter.h		2738	
136		threadpatternconverter.h		1830	
137		linelocationpatternconverter.h		1862	
138		levelpatternconverter.h		1886	

139	throwableinformationpatternconverter.h		2353	
140	lineseparatorpatternconverter.h		1965	
141	datepatternconverter.h		2480	
142	literalpatternconverter.h		1898	
143	namepatternconverter.h		2215	
144	nameabbreviator.h		2467	
145	classnamepatternconverter.h		1991	
146	filelocationpatternconverter.h		1884	
147	relativetimepatternconverter.h		1886	
148	patternconverter.h		3537	
149	propertiespatternconverter.h		2405	
150	integerpatternconverter.h		1807	
151	messagepatternconverter.h		1813	
152	asyncappender.h		10363	
153	portability.h		926	
154	dailyrollingfileappender.h		6499	
155	logmanager.h		7088	
156	appender.h		4571	
157	fileappender.h		8843	
158	consoleappender.h		2887	
159	propertysetter.h		4657	
160	nteventlogappender.h		4621	
161	outputdebugstringappender.h		1804	
162	writerappender.h		8255	
163	socketappender.h		5947	
164	syslogappender.h		6429	
165	telnetappender.h		6205	
166	socketappenderskeleton.h		6824	
167	sockethubappender.h		9015	
168	smtpappender.h		12128	
169	xmlsocketappender.h		6363	
170	logstring.h		2306	
171	simplelayout.h		2829	
172	level.h		10963	
173	fallbackerrorhandler.h		4642	
174	appenderskeleton.h		7552	
175	mdc.h		10342	
176	ttcclayout.h		7183	
177	propertyconfigurator.h		13543	
178	stream.h		20771	
179	logger.h		74966	
180	patternlayout.h		14232	
181	integer.cpp		1069	
182	locationinfo.cpp		5809	
183	properties.cpp		14490	

184	file.cpp		5977	
185	inputstreamreader.cpp		2362	
186	hierarchy.cpp		10880	
187	socket.cpp		3970	
188	socketappender.cpp		3302	
189	aprinitializer.cpp		1959	
190	propertiespatternconverter.cpp		2760	
191	ndc.cpp		7610	
192	threadpatternconverter.cpp		1748	
193	basicconfigurator.cpp		1655	
194	fulllocationpatternconverter.cpp		2056	
195	propertyconfigurator.cpp		16068	
196	nameabbreviator.cpp		8867	
197	objectoutputstream.cpp		6480	
198	outputstream.cpp		1046	
199	socketappenderskeleton.cpp		6239	
200	logstream.cpp		13956	
201	resourcebundle.cpp		3489	
202	relativetimepatternconverter.cpp		1891	
203	charsetencoder.cpp		18715	
204	inetaddress.cpp		4140	
205	throwableinformationpatternconverter.cpp		2275	
206	patternconverter.cpp		1604	
207	exception.cpp		11625	
208	rollingpolicybase.cpp		4609	
209	classregistration.cpp		1085	
210	stringmatchfilter.cpp		2232	
211	datagrampacket.cpp		2138	
212	datagramsocket.cpp		5266	
213	layout.cpp		1321	
214	threadlocal.cpp		1823	
215	levelmatchfilter.cpp		2338	
216	htmllayout.cpp		8432	
217	fallbackerrorhandler.cpp		3698	
218	writer.cpp		985	
219	charsetdecoder.cpp		15992	
220	obsoleteerollingfileappender.cpp		5102	
221	triggeringpolicy.cpp		1177	
222	fileoutputstream.cpp		2851	
223	patternparser.cpp		10130	
224	integerpatternconverter.cpp		1741	
225	systemerrwriter.cpp		1932	
226	transform.cpp		3138	
227	sizebasedtriggeringpolicy.cpp		1954	
228	onlyonceerrorhandler.cpp		2345	

229		outputstreamwriter.cpp		2425	
230		writerappender.cpp		6840	
231		datepatternconverter.cpp		4422	
232		class.cpp		6249	
233		telnetappender.cpp		8295	
234		rolloverdescription.cpp		1909	
235		nventlogappender.cpp		9785	
236		dateformat.cpp		1237	
237		filterbasedtriggeringpolicy.cpp		2658	
238		objectptr.cpp		1637	
239		timezone.cpp		7167	
240		systemoutwriter.cpp		1937	
241		patternlayout.cpp		6237	
242		asynccappender.cpp		11320	
243		objectimpl.cpp		1319	
244		filedatepatternconverter.cpp		1490	
245		sockethubappender.cpp		7798	
246		filerenameaction.cpp		1333	
247		fileinputstream.cpp		2678	
248		zipcompressaction.cpp		2934	
249		system.cpp		3083	
250		logmanager.cpp		5844	
251		fixedwindowrollingpolicy.cpp		8202	
252		stringhelper.cpp		4623	
253		filewatchdog.cpp		2600	
254		cacheddateformat.cpp		9785	
255		appenderattachableimpl.cpp		4072	
256		reader.cpp		985	
257		formattinginfo.cpp		2279	
258		odbcappender.cpp		8547	
259		methodlocationpatternconverter.cpp		1809	
260		bufferedwriter.cpp		1690	
261		bytearrayinputstream.cpp		1739	
262		strftimedateformat.cpp		1783	
263		syslogappender.cpp		11916	
264		rollingpolicy.cpp		927	
265		rollingfileappender.cpp		10423	
266		loggingevent.cpp		10395	
267		gzcompressaction.cpp		3364	
268		namepatternconverter.cpp		1965	
269		outputdebugstringappender.cpp		1501	
270		loggingeventpatternconverter.cpp		1644	
271		simpledateformat.cpp		19000	
272		filelocationpatternconverter.cpp		1810	
273		threadspecificdata.cpp		3643	

274	xmlsocketappender.cpp		3609	
275	inputstream.cpp		1040	
276	optionconverter.cpp		13129	
277	appenderskeleton.cpp		4552	
278	date.cpp		1419	
279	locale.cpp		1462	
280	propertysetter.cpp		3596	
281	synchronized.cpp		1586	
282	propertyresourcebundle.cpp		1752	
283	bytebuffer.cpp		1682	
284	literalpatternconverter.cpp		2073	
285	simplelayout.cpp		1338	
286	loggerpatternconverter.cpp		1939	
287	levelrangefilter.cpp		2877	
288	configurator.cpp		1054	
289	ttcclayout.cpp		2589	
290	syslogwriter.cpp		2171	
291	manualtriggeringpolicy.cpp		1543	
292	messagebuffer.cpp		14451	
293	level.cpp		6093	
294	threadcxx.cpp		4618	
295	cyclicbuffer.cpp		4029	
296	loglog.cpp		3273	
297	fileappender.cpp		9058	
298	mutex.cpp		1786	
299	stringtokenizer.cpp		1932	
300	filter.cpp		1347	
301	datelayout.cpp		4177	
302	action.cpp		1831	
303	defaultloggerfactory.cpp		1140	
304	loader.cpp		2434	
305	domconfigurator.cpp		39492	
306	levelpatternconverter.cpp		2568	
307	relativetimedateformat.cpp		1418	
308	serversocket.cpp		3891	
309	timebasedrollingpolicy.cpp		5555	
310	defaultconfigurator.cpp		4008	
311	smtpappender.cpp		20605	
312	rootlogger.cpp		1470	
313	pool.cpp		2212	
314	socketoutputstream.cpp		1725	
315	messagepatternconverter.cpp		1744	
316	transcoder.cpp		15115	
317	dailyrollingfileappender.cpp		3089	
318	defaultrepositoryselector.cpp		1342	

```

319 | condition.cpp | 1968 |
320 | lineseparatorpatternconverter.cpp | 1944 |
321 | classnamepatternconverter.cpp | 2016 |
322 | logger.cpp | 24986 |
323 | mdc.cpp | 6219 |
324 | xmllayout.cpp | 6325 |
325 | ndcpatternconverter.cpp | 1741 |
326 | bytearrayoutputstream.cpp | 1622 |
327 | consoleappender.cpp | 4133 |
328 | linelocationpatternconverter.cpp | 1866 |
329 +-----+-----+
330 325 rows in set (0.00 sec)

```

and for the C++ Sockets Library the output is

The Bytes for each File in the C++ Sockets Library

```

1 +-----+-----+
2 | name | Byte Count |
3 +-----+-----+
4 | Event.h | 2081 |
5 | Debug.cpp | 1816 |
6 | IHttpServer.h | 1917 |
7 | SocketStream.cpp | 1920 |
8 | HttpRequest.h | 3694 |
9 | EventTime.h | 2166 |
10 | HttpRequest.cpp | 9981 |
11 | HttpPostSocket.h | 3111 |
12 | Thread.cpp | 3364 |
13 | HttpBaseSocket.h | 2293 |
14 | MemFile.cpp | 6486 |
15 | Ajp13Socket.cpp | 9945 |
16 | HttpPostSocket.cpp | 7599 |
17 | XmlException.cpp | 1513 |
18 | SocketThread.cpp | 873 |
19 | EventHandler.h | 2421 |
20 | Exception.h | 1907 |
21 | IEventOwner.h | 2255 |
22 | Lock.h | 1606 |
23 | Sockets-config.cpp | 2807 |
24 | Ipv4Address.h | 3230 |
25 | ResolveServer.cpp | 2080 |
26 | StreamSocket.cpp | 2984 |
27 | SctpSocket.h | 3271 |
28 | HttpResponse.cpp | 6427 |
29 | EventHandler.cpp | 4420 |

```

30	SocketStream.h		1825	
31	SocketHandlerEp.cpp		4566	
32	UdpSocket.cpp		19767	
33	Mutex.cpp		1891	
34	UdpSocket.h		7419	
35	StdoutLog.cpp		2619	
36	ResolvSocket.cpp		9859	
37	StreamWriter.h		1868	
38	HttpdCookies.cpp		6842	
39	FileStream.h		1674	
40	File.cpp		3358	
41	Utility.cpp		34827	
42	XmlDocument.h		2043	
43	sockets-config.h		2588	
44	EventTime.cpp		2257	
45	HTTPSocket.cpp		11150	
46	SctpSocket.cpp		11375	
47	SocketThread.h		684	
48	socket_include.cpp		3669	
49	Parse.cpp		6235	
50	IFileUpload.h		1711	
51	AjpBaseSocket.h		2556	
52	Debug.h		1805	
53	HttpdForm.h		4010	
54	SocketHandlerThread.h		1737	
55	SSLInitializer.cpp		3829	
56	Ipv6Address.h		3354	
57	SocketHandlerThread.cpp		1905	
58	SmtpdSocket.cpp		4571	
59	XmlNode.h		4462	
60	Thread.h		3178	
61	Base64.cpp		7841	
62	SocketHandler.h		9329	
63	Lock.cpp		1485	
64	XmlException.h		1804	
65	HttpTransaction.cpp		5948	
66	FileStream.cpp		1679	
67	Base64.h		2318	
68	TcpSocket.cpp		38044	
69	StreamWriter.cpp		2190	
70	SocketHandler.cpp		29905	
71	Json.h		2120	
72	Mutex.h		1840	
73	Exception.cpp		1739	
74	IStream.h		1788	

75	HttpBaseSocket.cpp		5153	
76	HTTPSocket.h		5868	
77	IFile.h		2103	
78	ListenSocket.h		13161	
79	ResolvSocket.h		3051	
80	HttpResponse.h		2737	
81	Semaphore.h		2312	
82	IEventOwner.cpp		1915	
83	IBase.h		1056	
84	TcpSocket.h		11514	
85	Semaphore.cpp		2473	
86	XmlDocument.cpp		2465	
87	HttpDebugSocket.h		2162	
88	AjpBaseSocket.cpp		7789	
89	SSLInitializer.h		2276	
90	HttpDebugSocket.cpp		3457	
91	socket_include.h		6491	
92	SocketAddress.h		3045	
93	StdLog.h		1937	
94	HttpdSocket.h		3206	
95	HttpPutSocket.h		2445	
96	ISocketHandler.h		8880	
97	HttpdCookies.h		2662	
98	ResolvServer.h		2002	
99	Ipv6Address.cpp		5841	
100	XmlNode.cpp		6722	
101	MemFile.h		2963	
102	SmtpdSocket.h		4063	
103	HttpGetSocket.h		2038	
104	HttpClientSocket.h		4429	
105	Ajp13Socket.h		2260	
106	Event.cpp		1936	
107	HttpPutSocket.cpp		2880	
108	HttpdSocket.cpp		6740	
109	HttpTransaction.h		2957	
110	Ipv4Address.cpp		4682	
111	IEventHandler.h		2311	
112	Socket.h		23249	
113	HttpClientSocket.cpp		5855	
114	IMutex.h		1612	
115	Parse.h		2606	
116	Json.cpp		11771	
117	Socket.cpp		36527	
118	StreamSocket.h		4171	
119	StdoutLog.h		1821	

```

120 | HttpdForm.cpp          |      14870 |
121 | File.h                 |      2330 |
122 | ajp13.h                |      3254 |
123 | Utility.h              |      7510 |
124 | SocketHandlerEp.h     |      2622 |
125 | HttpGetSocket.cpp     |      3099 |
126 +-----+
127 122 rows in set (0.00 sec)

```

A-1.9 The Number of Functions

The output for the libodbc++ project for this query is

The Number of Functions in libodbc++

```

1 +-----+
2 | Function Count |
3 +-----+
4 |           1298 |
5 +-----+
6 1 row in set (0.00 sec)

```

for the log4cxx project the output is

The Number of Functions in log4cxx

```

1 +-----+
2 | Function Count |
3 +-----+
4 |           1680 |
5 +-----+
6 1 row in set (3.19 sec)

```

and for the C++ Sockets Library the output is

The Number of Functions in the C++ Sockets Library

```

1 +-----+
2 | Function Count |
3 +-----+
4 |           3331 |
5 +-----+
6 1 row in set (0.90 sec)

```

A-1.10 The Number of Functions per Generated XML File

The output for the libodbc++ project for this query is

The Functions per Generated XML File in libodbc++

```
1 +-----+-----+
2 | xml_id | Function Count per XML File |
3 +-----+-----+
4 |      1 |                2194 |
5 |      2 |                1834 |
6 |      3 |                1839 |
7 |      4 |                1052 |
8 |      5 |                1837 |
9 |      6 |                1840 |
10 |     7 |                2192 |
11 |     8 |                1272 |
12 |     9 |                1276 |
13 |    10 |                2554 |
14 |    11 |                1837 |
15 |    12 |                2193 |
16 |    13 |                2196 |
17 |    14 |                 917 |
18 |    16 |                 134 |
19 |    17 |                 134 |
20 |    18 |                2082 |
21 |    19 |                2436 |
22 |    20 |                2082 |
23 |    21 |                1043 |
24 |    22 |                2544 |
25 +-----+-----+
26 21 rows in set (0.00 sec)
```

for the log4cxx project the output is

The Functions per Generated XML File in log4cxx

```
1 +-----+-----+
2 | xml_id | Function Count per XML File |
3 +-----+-----+
4 |      1 |                1545 |
5 |      2 |                1790 |
6 |      3 |                2433 |
7 |      4 |                1331 |
8 |      5 |                1544 |
9 |      6 |                2200 |
10 |     7 |                1799 |
```

11		8		1544	
12		9		2003	
13		10		770	
14		11		1546	
15		12		2012	
16		13		1544	
17		14		770	
18		15		770	
19		16		770	
20		17		1544	
21		18		1544	
22		19		770	
23		20		770	
24		21		770	
25		22		770	
26		23		1540	
27		24		673	
28		25		1540	
29		26		667	
30		27		770	
31		28		667	
32		29		673	
33		30		1580	
34		31		770	
35		32		1540	
36		33		770	
37		34		1547	
38		35		2284	
39		36		1544	
40		37		2065	
41		38		673	
42		39		1338	
43		40		1703	
44		41		1344	
45		42		1540	
46		43		1880	
47		44		1565	
48		45		1542	
49		46		1581	
50		47		134	
51		48		1569	
52		49		1325	
53		50		1017	
54		51		1539	
55		52		1320	

56		53		1028	
57		54		667	
58		55		1445	
59		56		1338	
60		57		1201	
61		58		673	
62		59		1566	
63		60		1036	
64		61		1885	
65		62		1746	
66		63		667	
67		64		1818	
68		65		673	
69		66		801	
70		67		1540	
71		68		1933	
72		69		1338	
73		70		1544	
74		71		1547	
75		72		1570	
76		73		1782	
77		74		1314	
78		75		1323	
79		76		1891	
80		77		1465	
81		78		1856	
82		79		2061	
83		80		1338	
84		81		1885	
85		82		2071	
86		83		660	
87		84		1456	
88		85		1859	
89		86		1338	
90		87		1942	
91		88		667	
92		89		2061	
93		90		1819	
94		91		1350	
95		92		1543	
96		93		1669	
97		94		673	
98		95		1348	
99		97		667	
100		98		1514	

101		99		1577	
102		100		2071	
103		101		1254	
104		102		1456	
105		103		1400	
106		104		1676	
107		105		1863	
108		106		1837	
109		107		1338	
110		108		134	
111		109		1544	
112		110		1338	
113		111		1544	
114		112		1547	
115		113		1340	
116		114		1779	
117		115		1544	
118		116		770	
119		117		1544	
120		118		1544	
121		119		1544	
122		120		2084	
123		121		2084	
124		122		1547	
125		124		1545	
126		125		1443	
127		126		1545	
128		127		1443	
129		128		1545	
130		129		1339	
131		130		1545	
132		131		1544	
133		132		1545	
134		133		1545	
135		134		1545	
136		135		1546	
137		136		1545	
138		137		1544	
139		138		1545	
140		139		1544	
141		140		1351	
142		141		1545	
143		142		1545	
144		143		1545	
145		144		1350	

```

146 |      145 |                1545 |
147 |      146 |                1351 |
148 |      147 |                1545 |
149 |      148 |                2083 |
150 |      149 |                 134 |
151 |      150 |                1544 |
152 |      151 |                1965 |
153 |      152 |                 673 |
154 |      153 |                1664 |
155 |      154 |                1546 |
156 |      155 |                1430 |
157 |      156 |                 904 |
158 |      157 |                 904 |
159 |      158 |                1544 |
160 |      159 |                2069 |
161 |      160 |                1549 |
162 |      161 |                2092 |
163 |      162 |                1540 |
164 |      163 |                2069 |
165 |      164 |                1788 |
166 |      165 |                1776 |
167 |      166 |                 660 |
168 |      167 |                1544 |
169 |      168 |                1866 |
170 |      169 |                1544 |
171 |      170 |                1775 |
172 |      171 |                1336 |
173 |      172 |                1546 |
174 |      173 |                2012 |
175 |      174 |                 770 |
176 |      175 |                2061 |
177 |      176 |                1544 |
178 |      177 |                 770 |
179 |      178 |                 774 |
180 |      179 |                1192 |
181 +-----+
182 177 rows in set (3 min 20.29 sec)

```

and for the C++ Sockets Library the output is

The Functions per Generated XML File in the C++ Sockets Library

```

1 +-----+
2 | xml_id | Function Count per XML File |
3 +-----+
4 |      1 |                302 |

```

5		2		2126	
6		3		134	
7		4		1728	
8		5		2144	
9		6		302	
10		7		6390	
11		8		624	
12		9		6324	
13		10		1570	
14		11		6324	
15		12		268	
16		13		6324	
17		14		6338	
18		15		1890	
19		16		268	
20		17		268	
21		18		2026	
22		19		1068	
23		20		6310	
24		21		6310	
25		22		1890	
26		23		6324	
27		24		6310	
28		25		594	
29		26		1320	
30		27		6324	
31		28		1890	
32		29		1320	
33		30		1320	
34		31		1556	
35		32		1890	
36		33		268	
37		34		6324	
38		35		722	
39		36		1320	
40		37		660	
41		38		6324	
42		39		2124	
43		40		6324	
44		41		6152	
45		42		268	
46		43		6324	
47		44		1890	
48		45		1556	
49		46		6332	

```

50 |      47 |                1890 |
51 |      48 |                6324 |
52 |      49 |                2326 |
53 |      50 |                 660 |
54 |      51 |                 660 |
55 |      52 |                3163 |
56 |      53 |                 624 |
57 |      54 |                 134 |
58 |      55 |                6324 |
59 |      56 |                 864 |
60 |      57 |                 660 |
61 |      58 |                6324 |
62 |      59 |                6390 |
63 |      60 |                3162 |
64 |      61 |                6326 |
65 |      62 |                6324 |
66 |      63 |                 134 |
67 |      64 |               6568 |
68 |      65 |                 134 |
69 |      66 |                 134 |
70 +-----+
71 66 rows in set (0.00 sec)

```

A-1.11 Total Class Count

The output for the libodbc++ project for this query is

Total Class Count for libodbc++

```

1 +-----+
2 | Class Count |
3 +-----+
4 |           361 |
5 +-----+
6 1 row in set (0.05 sec)

```

for the log4cxx project the output is

Total Class Count for log4cxx

```
1 +-----+
2 | Class Count |
3 +-----+
4 |           752 |
5 +-----+
6 1 row in set (0.49 sec)
```

and for the C++ Sockets Library the output is

Total Class Count for the C++ Sockets Library

```
1 +-----+
2 | Class Count |
3 +-----+
4 |           280 |
5 +-----+
6 1 row in set (0.04 sec)
```

A-1.12 Total Variable Count

The output for the libodbc++ project for this query is

The Total Variable Count for libodbc++

```
1 +-----+
2 | Variable Count |
3 +-----+
4 |           392 |
5 +-----+
6 1 row in set (0.06 sec)
```

for the log4cxx project the output is

The Total Variable Count for log4cxx

```
1 +-----+
2 | Variable Count |
3 +-----+
4 |           1050 |
5 +-----+
6 1 row in set (0.61 sec)
```

and for the C++ Sockets Library the output is

The Project Variable Count for the C++ Sockets Library

```
1 +-----+
2 | Variable Count |
3 +-----+
4 |           285 |
5 +-----+
6 1 row in set (0.04 sec)
```

A-1.13 Total Private Variable Count

The output for the libodbc++ project for this query is

Total Private Variable Count for libodbc++

```
1 +-----+
2 | Private Variable Count |
3 +-----+
4 |                   81 |
5 +-----+
6 1 row in set (0.02 sec)
```

for the log4cxx project the output is

Total Private Variable Count for log4cxx

```
1 +-----+
2 | Private Variable Count |
3 +-----+
4 |                   23 |
5 +-----+
6 1 row in set (0.04 sec)
```

and for the C++ Sockets Library the output is

The Private Variable Count for the C++ Sockets Library

```
1 +-----+
2 | Private Variable Count |
3 +-----+
4 |                   40 |
5 +-----+
6 1 row in set (0.01 sec)
```

A-1.14 Total Protected Variable Count

The output for the libodbc++ project for this query is

Total Protected Variable Count for libodbc++

```
1 +-----+
2 | Protected Variable Count |
3 +-----+
4 |                               0 |
5 +-----+
6 1 row in set (0.01 sec)
```

for the log4cxx project the output is

Total Protected Variable Count for log4cxx

```
1 +-----+
2 | Protected Variable Count |
3 +-----+
4 |                               0 |
5 +-----+
6 1 row in set (0.03 sec)
```

and for the C++ Sockets Library the output is

Total Protected Variable Count for the C++ Sockets Library

```
1 +-----+
2 | Protected Variable Count |
3 +-----+
4 |                               1 |
5 +-----+
6 1 row in set (0.01 sec)
```

A-1.15 Total Public Variable Count

The output for the libodbc++ project for this query is

Total Public Variable Count for libodbc++

```
1 +-----+
2 | Public Variable Count |
3 +-----+
4 |                311 |
5 +-----+
6 1 row in set (0.06 sec)
```

for the log4cxx project the output is

Total Public Variable Count for log4cxx

```
1 +-----+
2 | Public Variable Count |
3 +-----+
4 |                1029 |
5 +-----+
6 1 row in set (0.59 sec)
```

and for the C++ Sockets Library the output is

Total Public Variable Count for the C++ Sockets Library

```
1 +-----+
2 | Public Variable Count |
3 +-----+
4 |                244 |
5 +-----+
6 1 row in set (0.05 sec)
```

A-1.16 List of Included Library Files

The output for the libodbc++ project for this query is

The List of Included Library Files for libodbc++

```
1 +-----+
2 | name |
3 +-----+
4 | /usr/include/time.h |
5 | /usr/include/string.h |
6 | /opt/projects/gccxml/install/share/gccxml-0.9/GCC/4.4/gccxml_builtins.h |
7 | /usr/include/sqlext.h |
8 | /usr/include/sqltypes.h |
9 | /usr/include/unistd.h |
```

```

10 | /usr/include/sql.h |
11 | /usr/include/pthread.h |
12 | /usr/include/bits/time.h |
13 | /usr/include/sqlucode.h |
14 | /usr/include/wchar.h |
15 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/gthr-default.h |
16 | /usr/include/stdio.h |
17 | /usr/include/stdlib.h |
18 | /usr/include/stdint.h |
19 | /usr/include/sched.h |
20 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/atomic_word.h |
21 | /usr/include/bits/pthreadtypes.h |
22 | /usr/include/c++/4.4/exception |
23 | /usr/include/wctype.h |
24 | /usr/include/c++/4.4/iosfwd |
25 | /usr/include/bits/types.h |
26 | /usr/include/ctype.h |
27 | /usr/include/sys/types.h |
28 | /usr/include/c++/4.4/new |
29 | /usr/include/bits/sigset.h |
30 | /usr/include/signal.h |
31 | /usr/include/libio.h |
32 | /usr/include/c++/4.4/bits/locale_facets.h |
33 | /usr/include/inttypes.h |
34 | /usr/include/c++/4.4/bits/locale_classes.h |
35 | /usr/include/_G_config.h |
36 | /usr/include/bits/waitstatus.h |
37 | /usr/include/bits/confname.h |
38 | /usr/include/assert.h |
39 | /usr/include/xlocale.h |
40 | /usr/include/sys/sysmacros.h |
41 | /usr/include/bits/sched.h |
42 | /usr/include/getopt.h |
43 | /usr/include/sys/select.h |
44 | /usr/include/bits/sys_errlist.h |
45 | /usr/include/locale.h |
46 | /usr/include/bits/setjmp.h |
47 | /usr/include/alloca.h |
48 | /usr/include/c++/4.4/bits/ios_base.h |
49 | /usr/lib/gcc/x86_64-linux-gnu/4.4.5/include/stddef.h |
50 | /usr/include/c++/4.4/cxxabi-forced.h |
51 | /usr/lib/gcc/x86_64-linux-gnu/4.4.5/include/stdarg.h |
52 | /usr/include/bits/locale.h |
53 | /usr/include/c++/4.4/bits/stl_bvector.h |
54 | /usr/include/c++/4.4/bits/functexcept.h |

```

```

55 | /usr/include/c++/4.4/bits/istream.tcc |
56 | /usr/include/c++/4.4/bits/stl_uninitialized.h |
57 | /usr/include/c++/4.4/bits/stringfwd.h |
58 | /usr/include/c++/4.4/bits/stl_iterator_base_types.h |
59 | /usr/include/c++/4.4/bits/localefwd.h |
60 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/c++locale.h |
61 | /usr/include/c++/4.4/bits/stl_algobase.h |
62 | /usr/include/c++/4.4/ostream |
63 | /usr/include/c++/4.4/bits/cpp_type_traits.h |
64 | /usr/include/c++/4.4/bits/locale_classes.tcc |
65 | /usr/include/c++/4.4/bits/postypes.h |
66 | /usr/include/c++/4.4/bits/stl_vector.h |
67 | /usr/include/c++/4.4/bits/stl_construct.h |
68 | /usr/include/c++/4.4/cstdlib |
69 | /usr/include/c++/4.4/bits/stl_tree.h |
70 | /usr/include/c++/4.4/bits/allocator.h |
71 | /usr/include/c++/4.4/streambuf |
72 | /usr/include/c++/4.4/bits/char_traits.h |
73 | /usr/include/c++/4.4/bits/basic_string.h |
74 | /usr/include/c++/4.4/bits/basic_string.tcc |
75 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/ctype_base.h |
76 | /usr/include/c++/4.4/bits/stl_iterator.h |
77 | /usr/include/c++/4.4/bits/ostream_insert.h |
78 | /usr/include/c++/4.4/bits/locale_facets.tcc |
79 | /usr/include/c++/4.4/bits/ostream.tcc |
80 | /usr/include/c++/4.4/istream |
81 | /usr/include/c++/4.4/bits/streambuf.tcc |
82 | <built-in> |
83 | /usr/include/c++/4.4/ext/numeric_traits.h |
84 | /usr/include/c++/4.4/ext/atomicity.h |
85 | /usr/include/c++/4.4/ext/type_traits.h |
86 | /usr/include/c++/4.4/ext/new_allocator.h |
87 | /usr/include/c++/4.4/sstream |
88 | /usr/include/c++/4.4/bits/sstream.tcc |
89 | /usr/include/c++/4.4/bits/vector.tcc |
90 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/ctype_inline.h |
91 | /usr/include/c++/4.4/bits/basic_ios.h |
92 | /usr/include/c++/4.4/bits/basic_ios.tcc |
93 | /usr/include/c++/4.4/bits/stl_pair.h |
94 | /usr/include/c++/4.4/bits/stl_function.h |
95 | /usr/include/c++/4.4/bits/stl_map.h |
96 | /usr/include/errno.h |
97 | /usr/include/bits/errno.h |
98 | /usr/include/c++/4.4/backward/auto_ptr.h |
99 | /usr/include/c++/4.4/bits/stl_set.h |

```

100	/usr/include/qt3/qframe.h	
101	/usr/include/qt3/qglobal.h	
102	/usr/include/qt3/qtextstream.h	
103	/usr/include/qt3/qmenudata.h	
104	/usr/include/qt3/qwindowdefs.h	
105	/usr/include/qt3/qdict.h	
106	/usr/include/qt3/qscrollview.h	
107	/usr/include/qt3/qpushbutton.h	
108	/usr/include/qt3/qpixmap.h	
109	/usr/include/qt3/qpopupmenu.h	
110	/usr/include/qt3/qpalette.h	
111	/usr/include/qt3/qlistview.h	
112	/usr/include/qt3/qcstring.h	
113	/usr/include/qt3/qobjectdefs.h	
114	/usr/include/qt3/qiconset.h	
115	/usr/include/qt3/qptrcollection.h	
116	/usr/include/qt3/qregion.h	
117	/usr/include/qt3/qevent.h	
118	/usr/include/qt3/qmultilineedit.h	
119	/usr/include/qt3/qfontmetrics.h	
120	/usr/include/qt3/qstrlist.h	
121	/usr/include/qt3/qvaluelist.h	
122	/usr/include/qt3/qdockwindow.h	
123	/usr/include/qt3/qmap.h	
124	/usr/include/qt3/qcolor.h	
125	/usr/include/qt3/qscrollbar.h	
126	/usr/include/qt3/qrect.h	
127	/usr/include/qt3/qsize.h	
128	/usr/include/qt3/qpoint.h	
129	/usr/include/qt3/qstring.h	
130	/usr/include/qt3/qkeysequence.h	
131	/usr/include/qt3/qvariant.h	
132	/usr/include/qt3/qpaintdevice.h	
133	/usr/include/qt3/qwidget.h	
134	/usr/include/qt3/qobject.h	
135	/usr/include/qt3/qgarray.h	
136	/usr/include/qt3/qtextedit.h	
137	/usr/include/qt3/qgvector.h	
138	/usr/include/qt3/qmainwindow.h	
139	/usr/include/qt3/qfont.h	
140	/usr/include/qt3/qbrush.h	
141	/usr/include/qt3/qstringlist.h	
142	/usr/include/qt3/qglist.h	
143	/usr/include/qt3/qbutton.h	
144	/usr/include/qt3/qptrlist.h	

```

145 | /usr/include/qt3/qfontinfo.h |
146 | /usr/include/qt3/qmime.h |
147 | /usr/include/qt3/qgdict.h |
148 | /usr/include/qt3/qdatastream.h |
149 | /usr/include/qt3/qnamespace.h |
150 | /usr/include/qt3/qpair.h |
151 | /usr/include/qt3/qrangecontrol.h |
152 | /usr/include/qt3/qmemarray.h |
153 | /usr/include/qt3/qshared.h |
154 | /usr/include/qt3/qmenubar.h |
155 | /usr/include/qt3/qptrvector.h |
156 | /usr/include/qt3/qsizepolicy.h |
157 | /usr/include/qt3/qtoolbar.h |
158 | /usr/include/qt3/qiodevice.h |
159 | /usr/include/qt3/qstylesheet.h |
160 | /usr/include/qt3/qlabel.h |
161 | /usr/include/qt3/qsignal.h |
162 | /usr/include/c++/4.4/bits/stl_list.h |
163 | /usr/include/c++/4.4/bits/list.tcc |
164 | /usr/include/qt3/qlayout.h |
165 | /usr/include/qt3/qdialog.h |
166 | /usr/include/qt3/qapplication.h |
167 | /usr/include/qt3/qtranslator.h |
168 | /usr/include/qt3/qasciidict.h |
169 | /usr/include/qt3/qdesktopwidget.h |
170 | /usr/include/qt3/qsemimodal.h |
171 | /usr/include/qt3/qlineedit.h |
172 | /usr/include/qt3/qcombobox.h |
173 | /usr/include/c++/4.4/stdexcept |
174 +-----+
175 170 rows in set (1.21 sec)

```

for the log4cxx project the output is

The List of Included Library Files for log4cxx

```

1 +-----+
2 | name |
3 +-----+
4 | /usr/include/time.h |
5 | /opt/projects/gccxml/install/share/gccxml-0.9/GCC/4.4/gccxml_builtins.h |
6 | /usr/include/unistd.h |
7 | /usr/include/pthread.h |
8 | /usr/include/wctype.h |
9 | /usr/include/wchar.h |
10 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/gthr-default.h |

```

```

11 | /usr/include/sched.h |
12 | /usr/include/c++/4.4/bits/locale_facets.h |
13 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/atomic_word.h |
14 | /usr/include/bits/pthreadtypes.h |
15 | /usr/include/c++/4.4/iosfwd |
16 | /usr/include/bits/types.h |
17 | /usr/include/ctype.h |
18 | /usr/include/c++/4.4/new |
19 | /usr/include/bits/sigset.h |
20 | /usr/include/signal.h |
21 | /usr/include/c++/4.4/exception |
22 | /usr/include/stdio.h |
23 | /usr/include/c++/4.4/bits/locale_classes.h |
24 | /usr/include/bits/confname.h |
25 | /usr/include/xlocale.h |
26 | /usr/include/c++/4.4/cxxabi-forced.h |
27 | /usr/include/bits/sched.h |
28 | /usr/include/getopt.h |
29 | /usr/include/locale.h |
30 | /usr/include/bits/setjmp.h |
31 | /usr/include/c++/4.4/bits/ios_base.h |
32 | /usr/lib/gcc/x86_64-linux-gnu/4.4.5/include/stddef.h |
33 | /usr/lib/gcc/x86_64-linux-gnu/4.4.5/include/stdarg.h |
34 | /usr/include/bits/locale.h |
35 | /usr/include/c++/4.4/bits/stl_bvector.h |
36 | /usr/include/c++/4.4/bits/functexcept.h |
37 | /usr/include/c++/4.4/bits/istream.tcc |
38 | /usr/include/c++/4.4/bits/stl_uninitialized.h |
39 | /usr/include/c++/4.4/bits/stringfwd.h |
40 | /usr/include/c++/4.4/bits/stl_pair.h |
41 | /usr/include/c++/4.4/bits/stl_iterator_base_types.h |
42 | /usr/include/c++/4.4/bits/localefwd.h |
43 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/c++locale.h |
44 | /usr/include/c++/4.4/bits/stl_algobase.h |
45 | /usr/include/c++/4.4/ostream |
46 | /usr/include/c++/4.4/bits/cpp_type_traits.h |
47 | /usr/include/c++/4.4/bits/stl_vector.h |
48 | /usr/include/c++/4.4/bits/locale_classes.tcc |
49 | /usr/include/c++/4.4/bits/postypes.h |
50 | /usr/include/c++/4.4/bits/stl_construct.h |
51 | /usr/include/c++/4.4/bits/stl_tree.h |
52 | /usr/include/c++/4.4/bits/allocator.h |
53 | /usr/include/c++/4.4/streambuf |
54 | /usr/include/c++/4.4/bits/char_traits.h |
55 | /usr/include/c++/4.4/bits/stl_map.h |

```

```

56 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/ctype_base.h |
57 | /usr/include/c++/4.4/bits/stl_iterator.h |
58 | /usr/include/c++/4.4/bits/ostream_insert.h |
59 | /usr/include/c++/4.4/bits/locale_facets.tcc |
60 | /usr/include/c++/4.4/bits/basic_string.h |
61 | /usr/include/c++/4.4/bits/streambuf.tcc |
62 | /usr/include/c++/4.4/bits/ostream.tcc |
63 | /usr/include/c++/4.4/istream |
64 | /usr/include/c++/4.4/bits/basic_string.tcc |
65 | /usr/include/c++/4.4/bits/stl_function.h |
66 | <built-in> |
67 | /usr/include/c++/4.4/ext/numeric_traits.h |
68 | /usr/include/c++/4.4/ext/atomicity.h |
69 | /usr/include/c++/4.4/ext/type_traits.h |
70 | /usr/include/c++/4.4/ext/new_allocator.h |
71 | /usr/include/c++/4.4/sstream |
72 | /usr/include/c++/4.4/bits/vector.tcc |
73 | /usr/include/c++/4.4/bits/sstream.tcc |
74 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/ctype_inline.h |
75 | /usr/include/c++/4.4/bits/basic_ios.h |
76 | /usr/include/c++/4.4/bits/basic_ios.tcc |
77 | /usr/include/c++/4.4/bits/stl_list.h |
78 | /usr/include/c++/4.4/bits/list.tcc |
79 | /usr/include/c++/4.4/bits/stl_deque.h |
80 | /usr/include/c++/4.4/bits/stl_stack.h |
81 | /usr/include/c++/4.4/bits/deque.tcc |
82 | /usr/include/libio.h |
83 | /usr/include/_G_config.h |
84 | /usr/include/bits/sys_errlist.h |
85 | /usr/include/string.h |
86 | /usr/include/apr-1.0/apr.h |
87 | /usr/include/bits/time.h |
88 | /usr/include/sys/socket.h |
89 | /usr/include/stdint.h |
90 | /usr/include/apr-1.0/apr_allocator.h |
91 | /usr/include/apr-1.0/apr_pools.h |
92 | /usr/include/sys/ucontext.h |
93 | /usr/include/sys/resource.h |
94 | /usr/include/bits/resource.h |
95 | /usr/include/bits/siginfo.h |
96 | /usr/include/bits/sigcontext.h |
97 | /usr/include/sys/types.h |
98 | /usr/include/apr-1.0/apr_general.h |
99 | /usr/include/apr-1.0/apr_strings.h |
100 | /usr/include/sys/uio.h |

```

```

101 | /usr/include/sys/select.h |
102 | /usr/include/errno.h |
103 | /usr/include/apr-1.0/apr_thread_mutex.h |
104 | /usr/include/bits/sigthread.h |
105 | /usr/include/bits/uio.h |
106 | /usr/include/sys/sysmacros.h |
107 | /usr/include/bits/sigstack.h |
108 | /usr/include/bits/socket.h |
109 | /usr/include/apr-1.0/apr_errno.h |
110 | /usr/include/bits/sigaction.h |
111 | /usr/include/sys/wait.h |
112 | /usr/include/bits/errno.h |
113 | /usr/include/bits/waitstatus.h |
114 | /usr/include/bits/sockaddr.h |
115 | /usr/include/apr-1.0/apr_file_info.h |
116 | /usr/include/apr-1.0/apr_file_io.h |
117 | /usr/include/apr-1.0/apr_time.h |
118 | /usr/include/apr-1.0/apr_tables.h |
119 | /usr/include/apr-1.0/apr_user.h |
120 | /usr/include/assert.h |
121 | /usr/include/apr-1.0/apr_atomic.h |
122 | /usr/include/apr-1.0/apr_thread_proc.h |
123 | /usr/include/stdlib.h |
124 | /usr/include/sys/time.h |
125 | /usr/include/alloca.h |
126 | /usr/include/c++/4.4/bits/stl_algo.h |
127 | /usr/include/c++/4.4/cstdlib |
128 | /usr/include/apr-1.0/apr_signal.h |
129 | /usr/include/apr-1.0/apr_network_io.h |
130 | /usr/include/netinet/in.h |
131 | /usr/include/bits/in.h |
132 | /usr/include/sys/stat.h |
133 | /usr/include/apr-1.0/apr_portable.h |
134 | /usr/include/apr-1.0/apr_xlate.h |
135 | /usr/include/apr-1.0/apr_shm.h |
136 | /usr/include/apr-1.0/apr_dso.h |
137 | /usr/include/bits/stat.h |
138 | /usr/include/dirent.h |
139 | /usr/include/apr-1.0/apr_global_mutex.h |
140 | /usr/include/bits/fcntl.h |
141 | /usr/include/apr-1.0/apr_proc_mutex.h |
142 | /usr/include/fcntl.h |
143 | /usr/include/bits/dirent.h |
144 | /usr/include/apr-1.0/apr_lib.h |
145 | /usr/include/apr-1.0/apr_thread_cond.h |

```



```

146 | /usr/include/apr-1.0/apr_env.h |
147 | /usr/include/c++/4.4/stdexcept |
148 | /usr/include/c++/4.4/limits |
149 | /usr/include/sys/syslog.h |
150 | /usr/include/c++/4.4/bits/locale_facets_nonio.h |
151 | /usr/include/c++/4.4/bits/codecvvt.h |
152 | /usr/include/libintl.h |
153 | /usr/include/c++/4.4/bits/locale_facets_nonio.tcc |
154 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/messages_members.h |
155 | /usr/include/c++/4.4/bits/streambuf_iterator.h |
156 | /usr/include/c++/4.4/iostream |
157 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/basic_file.h |
158 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/c++io.h |
159 | /usr/include/c++/4.4/fstream |
160 | /usr/include/c++/4.4/bits/fstream.tcc |
161 | /usr/include/apr-1.0/apr_xml.h |
162 | /usr/include/apr-1.0/apr_poll.h |
163 +-----+
164 159 rows in set (1 min 0.30 sec)

```

and for the C++ Sockets Library the output is

The List of Included Library Files for the C++ Sockets Library

```

1 +-----+
2 | name |
3 +-----+
4 | /opt/projects/gccxml/install/share/gccxml-0.9/GCC/4.4/gccxml_builtins.h |
5 | /usr/include/bits/time.h |
6 | /usr/include/stdint.h |
7 | /usr/include/bits/types.h |
8 | /usr/include/bits/sigset.h |
9 | /usr/include/sys/select.h |
10 | /usr/include/sys/time.h |
11 | /usr/include/time.h |
12 | /usr/include/inttypes.h |
13 | /usr/include/string.h |
14 | /usr/include/stdio.h |
15 | /usr/include/pthread.h |
16 | /usr/include/unistd.h |
17 | /usr/include/wchar.h |
18 | /usr/include/sys/socket.h |
19 | /usr/include/stdlib.h |
20 | /usr/include/netinet/in.h |
21 | /usr/include/sched.h |
22 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/atomic_word.h |

```

```

23 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/gthr-default.h |
24 | /usr/include/bits/pthreadtypes.h |
25 | /usr/include/ctype.h |
26 | /usr/include/c++/4.4/new |
27 | /usr/include/libio.h |
28 | /usr/include/sys/types.h |
29 | /usr/include/c++/4.4/exception |
30 | /usr/include/sys/uio.h |
31 | /usr/include/_G_config.h |
32 | /usr/include/bits/setjmp.h |
33 | /usr/include/bits/in.h |
34 | /usr/include/bits/socket.h |
35 | /usr/include/arpa/inet.h |
36 | /usr/include/bits/uio.h |
37 | /usr/include/xlocale.h |
38 | /usr/include/c++/4.4/bits/stl_list.h |
39 | /usr/include/c++/4.4/cxxabi-forced.h |
40 | /usr/include/sys/sysmacros.h |
41 | /usr/include/bits/sched.h |
42 | /usr/include/bits/sys_errlist.h |
43 | /usr/include/locale.h |
44 | /usr/include/getopt.h |
45 | /usr/include/alloca.h |
46 | /usr/lib/gcc/x86_64-linux-gnu/4.4.5/include/stddef.h |
47 | /usr/include/bits/confname.h |
48 | /usr/include/bits/waitstatus.h |
49 | /usr/lib/gcc/x86_64-linux-gnu/4.4.5/include/stdarg.h |
50 | /usr/include/bits/locale.h |
51 | /usr/include/bits/sockaddr.h |
52 | /usr/include/c++/4.4/bits/functexcept.h |
53 | /usr/include/c++/4.4/bits/stl_uninitialized.h |
54 | /usr/include/c++/4.4/iosfwd |
55 | /usr/include/c++/4.4/bits/stringfwd.h |
56 | /usr/include/c++/4.4/bits/stl_pair.h |
57 | /usr/include/c++/4.4/bits/cpp_type_traits.h |
58 | /usr/include/c++/4.4/bits/stl_iterator_base_types.h |
59 | /usr/include/c++/4.4/bits/localefwd.h |
60 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/c++locale.h |
61 | /usr/include/c++/4.4/bits/stl_algobase.h |
62 | /usr/include/c++/4.4/bits/stl_construct.h |
63 | /usr/include/c++/4.4/bits/stl_tree.h |
64 | /usr/include/c++/4.4/bits/stl_function.h |
65 | /usr/include/c++/4.4/bits/postypes.h |
66 | /usr/include/c++/4.4/bits/allocator.h |
67 | /usr/include/c++/4.4/bits/char_traits.h |

```

```

68 | /usr/include/c++/4.4/bits/stl_map.h |
69 | /usr/include/c++/4.4/backward/auto_ptr.h |
70 | /usr/include/c++/4.4/bits/stl_iterator.h |
71 | /usr/include/c++/4.4/bits/ostream_insert.h |
72 | /usr/include/c++/4.4/bits/basic_string.h |
73 | /usr/include/c++/4.4/bits/basic_string.tcc |
74 | <built-in> |
75 | /usr/include/c++/4.4/ext/numeric_traits.h |
76 | /usr/include/c++/4.4/ext/atomicity.h |
77 | /usr/include/c++/4.4/ext/type_traits.h |
78 | /usr/include/c++/4.4/ext/new_allocator.h |
79 | /usr/include/signal.h |
80 | /usr/include/c++/4.4/bits/stl_bvector.h |
81 | /usr/include/c++/4.4/bits/stl_vector.h |
82 | /usr/include/c++/4.4/bits/list.tcc |
83 | /usr/include/c++/4.4/bits/vector.tcc |
84 | /usr/include/openssl/bio.h |
85 | /usr/include/openssl/asn1.h |
86 | /usr/include/sys/stat.h |
87 | /usr/include/openssl/ssl.h |
88 | /usr/include/openssl/x509.h |
89 | /usr/include/openssl/ec.h |
90 | /usr/include/openssl/bn.h |
91 | /usr/include/openssl/pkcs7.h |
92 | /usr/include/openssl/openssl_typ.h |
93 | /usr/include/openssl/evp.h |
94 | /usr/include/openssl/stack.h |
95 | /usr/include/openssl/pem.h |
96 | /usr/include/openssl/tls1.h |
97 | /usr/include/openssl/pqueue.h |
98 | /usr/include/openssl/crypto.h |
99 | /usr/include/openssl/x509_vfy.h |
100 | /usr/include/openssl/rsa.h |
101 | /usr/include/openssl/lhash.h |
102 | /usr/include/semaphore.h |
103 | /usr/include/openssl/objects.h |
104 | /usr/include/openssl/dtls1.h |
105 | /usr/include/openssl/ecdsa.h |
106 | /usr/include/openssl/dsa.h |
107 | /usr/include/errno.h |
108 | /usr/include/openssl/ssl2.h |
109 | /usr/include/openssl/dh.h |
110 | /usr/include/openssl/ecdh.h |
111 | /usr/include/openssl/sha.h |
112 | /usr/include/openssl/buffer.h |

```

```

113 | /usr/include/openssl/ssl3.h |
114 | /usr/include/openssl/comp.h |
115 | /usr/include/bits/errno.h |
116 | /usr/include/bits/semaphore.h |
117 | /usr/include/bits/stat.h |
118 | /usr/include/openssl/hmac.h |
119 | /usr/include/netdb.h |
120 | /usr/include/rpc/netdb.h |
121 | /usr/include/bits/siginfo.h |
122 | /usr/include/bits/netdb.h |
123 | /usr/include/wctype.h |
124 | /usr/include/c++/4.4/bits/locale_facets.h |
125 | /usr/include/c++/4.4/bits/locale_classes.h |
126 | /usr/include/c++/4.4/bits/ios_base.h |
127 | /usr/include/c++/4.4/bits/istream.tcc |
128 | /usr/include/c++/4.4/ostream |
129 | /usr/include/c++/4.4/bits/locale_classes.tcc |
130 | /usr/include/c++/4.4/streambuf |
131 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/ctype_base.h |
132 | /usr/include/c++/4.4/bits/locale_facets.tcc |
133 | /usr/include/c++/4.4/bits/streambuf.tcc |
134 | /usr/include/c++/4.4/bits/ostream.tcc |
135 | /usr/include/c++/4.4/istream |
136 | /usr/include/c++/4.4/sstream |
137 | /usr/include/c++/4.4/bits/sstream.tcc |
138 | /usr/include/c++/4.4/x86_64-linux-gnu/bits/ctype_inline.h |
139 | /usr/include/c++/4.4/bits/basic_ios.h |
140 | /usr/include/c++/4.4/bits/basic_ios.tcc |
141 | /usr/include/fcntl.h |
142 | /usr/include/bits/fcntl.h |
143 +-----+
144 139 rows in set (10.68 sec)

```

A-1.17 Number of Included Library Files

The output for the libodbc++ project for this query is

Number of Included Library Files for libodbc++

```
1 +-----+
2 | Number of Included Library Files |
3 +-----+
4 |                               170 |
5 +-----+
6 1 row in set (0.97 sec)
```

for the log4cxx project the output is

Number of Included Library Files for log4cxx

```
1 +-----+
2 | Number of Included Library Files |
3 +-----+
4 |                               159 |
5 +-----+
6 1 row in set (1 min 6.33 sec)
```

and for the C++ Sockets Library the output is

Number of Included Library Files for the C++ Sockets Library

```
1 +-----+
2 | Number of Included Library Files |
3 +-----+
4 |                               139 |
5 +-----+
6 1 row in set (10.41 sec)
```

A-1.18 File Weight

The output for the libodbc++ project for this query is

The Individual File Weights for libodbc++

```
1 +-----+-----+-----+
2 | id | name                | File Weight |
3 +-----+-----+-----+
4 |  1 | statement.cpp       |           1 |
5 |  2 | driverinfo.h        |           7 |
6 |  3 | resultsetmetadata.cpp |           1 |
7 |  4 | threads.cpp         |           0 |
8 |  5 | errorhandler.cpp    |           1 |
9 |  6 | drivermanager.cpp   |           1 |
```

```

10 | 7 | datastream.cpp | 0 |
11 | 8 | callablestatement.cpp | 1 |
12 | 9 | dtconv.h | 10 |
13 | 10 | datetime.cpp | 1 |
14 | 11 | datahandler.h | 4 |
15 | 12 | connection.cpp | 1 |
16 | 13 | preparedstatement.cpp | 1 |
17 | 14 | datahandler.cpp | 2 |
18 | 15 | databasemetadata.cpp | 1 |
19 | 16 | datastream.h | 0 |
20 | 17 | driverinfo.cpp | 2 |
21 | 18 | resultset.cpp | 0 |
22 | 19 | types.h | 28 |
23 | 20 | resultsetmetadata.h | 6 |
24 | 21 | config-win32.h | 0 |
25 | 22 | statement.h | 14 |
26 | 23 | drivermanager.h | 4 |
27 | 24 | databasemetadata.h | 7 |
28 | 25 | threads.h | 0 |
29 | 26 | callablestatement.h | 3 |
30 | 27 | errorhandler.h | 21 |
31 | 28 | config.h | 0 |
32 | 29 | connection.h | 19 |
33 | 30 | preparedstatement.h | 7 |
34 | 31 | resultset.h | 9 |
35 | 32 | setup.h | 0 |
36 | 33 | mainwindow.cpp | 0 |
37 | 34 | resultwindow.h | 2 |
38 | 35 | connectwindow.cpp | 0 |
39 | 36 | connectwindow.h | 2 |
40 | 37 | resultwindow.cpp | 2 |
41 | 38 | main.cpp | 1 |
42 | 39 | mainwindow.h | 3 |
43 | 40 | isql++.cpp | 0 |
44 | 41 | isql++.h | 2 |
45 | 42 | config.h | 0 |
46 +-----+-----+-----+-----+
47 42 rows in set (0.11 sec)

```

for the log4cxx project the output is

The Individual File Weights for log4cxx

```

1 +-----+-----+-----+-----+
2 | id | name | File Weight |
3 +-----+-----+-----+

```

4		1		defaultconfigurator.h		3	
5		2		odbcappender.h		3	
6		3		rollingfileappender.h		3	
7		4		ndc.h		9	
8		5		layout.h		56	
9		6		hierarchy.h		3	
10		7		htmlayout.h		3	
11		8		basicconfigurator.h		2	
12		9		file.h		40	
13		10		provisionnode.h		4	
14		11		xmlayout.h		4	
15		12		domconfigurator.h		4	
16		13		levelrangefilter.h		3	
17		14		andfilter.h		1	
18		15		mapfilter.h		1	
19		16		expressionfilter.h		1	
20		17		stringmatchfilter.h		3	
21		18		levelmatchfilter.h		3	
22		19		propertyfilter.h		1	
23		20		denyallfilter.h		2	
24		21		locationinfofilter.h		1	
25		22		loggerfactory.h		174	
26		23		rootlogger.h		4	
27		24		hierarchyeventlistener.h		175	
28		25		defaultrepositoryselector.h		3	
29		26		optionhandler.h		178	
30		27		triggeringeventevaluator.h		6	
31		28		repositoryselector.h		20	
32		29		appenderattachable.h		176	
33		30		locationinfo.h		176	
34		31		errorhandler.h		45	
35		32		filter.h		57	
36		33		loggerrepository.h		174	
37		34		configurator.h		13	
38		35		loggingevent.h		134	
39		36		defaultloggerfactory.h		5	
40		37		filewatchdog.h		4	
41		38		iso8601dateformat.h		7	
42		39		outputstream.h		186	
43		40		stringhelper.h		54	
44		41		dateformat.h		26	
45		42		datelayout.h		5	
46		43		inetaddress.h		26	
47		44		optionconverter.h		27	
48		45		bytearrayoutputstream.h		3	

49		46		appenderattachableimpl.h		175	
50		47		strictmath.h		1	
51		48		cacheddateformat.h		7	
52		49		transform.h		5	
53		50		synchronized.h		23	
54		51		class.h		272	
55		52		locale.h		3	
56		53		objectptr.h		276	
57		54		xml.h		2	
58		55		resourcebundle.h		178	
59		56		reader.h		4	
60		57		aprinitializer.h		13	
61		58		socketoutputstream.h		4	
62		59		strftimedateformat.h		3	
63		60		condition.h		5	
64		61		datagramsocket.h		7	
65		62		simpledateformat.h		11	
66		63		bufferedoutputstream.h		1	
67		64		timezone.h		28	
68		65		absolutetimedateformat.h		3	
69		66		classregistration.h		272	
70		67		properties.h		11	
71		68		bytearrayinputstream.h		2	
72		69		bufferedwriter.h		3	
73		70		onlyonceerrorhandler.h		3	
74		71		propertyresourcebundle.h		3	
75		72		date.h		8	
76		73		system.h		4	
77		74		mutex.h		193	
78		75		stringtokenizer.h		4	
79		76		serversocket.h		6	
80		77		outputstreamwriter.h		19	
81		78		threadspecificdata.h		6	
82		79		fileoutputstream.h		4	
83		80		inputstream.h		20	
84		81		socket.h		16	
85		82		charsetdecoder.h		9	
86		83		thread.h		20	
87		84		systemerrwriter.h		4	
88		85		objectimpl.h		262	
89		86		integer.h		4	
90		87		loglog.h		41	
91		88		object.h		270	
92		89		fileinputstream.h		4	
93		90		transcoder.h		306	

94		91		datagrampacket.h		9	
95		92		cyclicbuffer.h		6	
96		93		relativetimedateformat.h		3	
97		94		datetimedateformat.h		3	
98		95		syslogwriter.h		5	
99		96		tchar.h		0	
100		97		inputstreamreader.h		2	
101		98		messagebuffer.h		176	
102		99		loader.h		5	
103		100		charsetencoder.h		183	
104		101		bytebuffer.h		17	
105		102		systemoutwriter.h		3	
106		103		exception.h		100	
107		104		objectoutputstream.h		178	
108		105		pool.h		221	
109		106		threadlocal.h		3	
110		107		writer.h		29	
111		108		log4cxx.h		317	
112		109		rollingpolicy.h		19	
113		110		rollingfileappender.h		5	
114		111		rolloverdescription.h		21	
115		112		filterbasedtriggeringpolicy.h		4	
116		113		rollingpolicybase.h		10	
117		114		action.h		29	
118		115		timebasedrollingpolicy.h		4	
119		116		triggeringpolicy.h		21	
120		117		rollingfileappenderskeleton.h		9	
121		118		manualtriggeringpolicy.h		4	
122		119		filerenameaction.h		4	
123		120		sizebasedtriggeringpolicy.h		4	
124		121		gzcompressaction.h		4	
125		122		zipcompressaction.h		4	
126		123		fixedwindowrollingpolicy.h		5	
127		124		log4cxx_private.h		0	
128		125		methodlocationpatternconverter.h		3	
129		126		patternparser.h		16	
130		127		ndcpatternconverter.h		3	
131		128		filedatepatternconverter.h		4	
132		129		loggerpatternconverter.h		3	
133		130		formattinginfo.h		18	
134		131		fulllocationpatternconverter.h		3	
135		132		loggingeventpatternconverter.h		44	
136		133		threadpatternconverter.h		3	
137		134		linelocationpatternconverter.h		3	
138		135		levelpatternconverter.h		3	

139		136		throwableinformationpatternconverter.h		3	
140		137		lineseparatorpatternconverter.h		3	
141		138		datepatternconverter.h		5	
142		139		literalpatternconverter.h		4	
143		140		namepatternconverter.h		7	
144		141		nameabbreviator.h		9	
145		142		classnamepatternconverter.h		3	
146		143		filelocationpatternconverter.h		3	
147		144		relativetimepatternconverter.h		3	
148		145		patternconverter.h		58	
149		146		propertiespatternconverter.h		3	
150		147		integerpatternconverter.h		5	
151		148		messagepatternconverter.h		3	
152		149		asynccappender.h		3	
153		150		portability.h		0	
154		151		dailyrollingfileappender.h		3	
155		152		logmanager.h		16	
156		153		appender.h		177	
157		154		fileappender.h		11	
158		155		consoleappender.h		4	
159		156		propertysetter.h		7	
160		157		nteventlogappender.h		1	
161		158		outputdebugstringappender.h		1	
162		159		writerappender.h		16	
163		160		socketappender.h		3	
164		161		syslogappender.h		3	
165		162		telnetappender.h		3	
166		163		socketappenderskeleton.h		7	
167		164		sockethubappender.h		3	
168		165		smtpappender.h		4	
169		166		xmlsocketappender.h		3	
170		167		logstring.h		306	
171		168		simplelayout.h		3	
172		169		level.h		176	
173		170		fallbackerrorhandler.h		2	
174		171		appenderskeleton.h		39	
175		172		mdc.h		184	
176		173		ttcclayout.h		3	
177		174		propertyconfigurator.h		4	
178		175		stream.h		2	
179		176		logger.h		174	
180		177		patternlayout.h		5	
181		178		integer.cpp		1	
182		179		locationinfo.cpp		1	
183		180		properties.cpp		1	

184		181		file.cpp		1	
185		182		inputstreamreader.cpp		0	
186		183		hierarchy.cpp		1	
187		184		socket.cpp		1	
188		185		socketappender.cpp		1	
189		186		aprinitializer.cpp		1	
190		187		propertiespatternconverter.cpp		1	
191		188		ndc.cpp		1	
192		189		threadpatternconverter.cpp		1	
193		190		basicconfigurator.cpp		1	
194		191		fulllocationpatternconverter.cpp		1	
195		192		propertyconfigurator.cpp		1	
196		193		nameabbreviator.cpp		1	
197		194		objectoutputstream.cpp		1	
198		195		outputstream.cpp		1	
199		196		socketappenderskeleton.cpp		1	
200		197		logstream.cpp		1	
201		198		resourcebundle.cpp		1	
202		199		relativetimepatternconverter.cpp		1	
203		200		charsetencoder.cpp		1	
204		201		inetaddress.cpp		1	
205		202		throwableinformationpatternconverter.cpp		1	
206		203		patternconverter.cpp		1	
207		204		exception.cpp		1	
208		205		rollingpolicybase.cpp		1	
209		206		classregistration.cpp		1	
210		207		stringmatchfilter.cpp		1	
211		208		datagrampacket.cpp		1	
212		209		datagramsocket.cpp		1	
213		210		layout.cpp		1	
214		211		threadlocal.cpp		1	
215		212		levelmatchfilter.cpp		1	
216		213		htmlayout.cpp		1	
217		214		fallbackerrorhandler.cpp		1	
218		215		writer.cpp		1	
219		216		charsetdecoder.cpp		1	
220		217		obsoleterollingfileappender.cpp		1	
221		218		triggeringpolicy.cpp		1	
222		219		fileoutputstream.cpp		1	
223		220		patternparser.cpp		1	
224		221		integerpatternconverter.cpp		1	
225		222		systemerrwriter.cpp		1	
226		223		transform.cpp		1	
227		224		sizebasedtriggeringpolicy.cpp		1	
228		225		onlyonceerrorhandler.cpp		1	

229		226		outputstreamwriter.cpp		1	
230		227		writerappender.cpp		1	
231		228		datepatternconverter.cpp		1	
232		229		class.cpp		1	
233		230		telnetappender.cpp		1	
234		231		rolloverdescription.cpp		1	
235		232		nventlogappender.cpp		0	
236		233		dateformat.cpp		1	
237		234		filterbasedtriggeringpolicy.cpp		1	
238		235		objectptr.cpp		1	
239		236		timezone.cpp		1	
240		237		systemoutwriter.cpp		1	
241		238		patternlayout.cpp		1	
242		239		asynccappender.cpp		1	
243		240		objectimpl.cpp		1	
244		241		filedatepatternconverter.cpp		1	
245		242		sockethubappender.cpp		1	
246		243		filerenameaction.cpp		1	
247		244		fileinputstream.cpp		1	
248		245		zipcompressaction.cpp		1	
249		246		system.cpp		1	
250		247		logmanager.cpp		1	
251		248		fixedwindowrollingpolicy.cpp		1	
252		249		stringhelper.cpp		1	
253		250		filewatchdog.cpp		1	
254		251		cacheddateformat.cpp		1	
255		252		appenderattachableimpl.cpp		1	
256		253		reader.cpp		1	
257		254		formattinginfo.cpp		1	
258		255		odbcappender.cpp		1	
259		256		methodlocationpatternconverter.cpp		1	
260		257		bufferedwriter.cpp		1	
261		258		bytearrayinputstream.cpp		1	
262		259		strftimedateformat.cpp		1	
263		260		syslogappender.cpp		1	
264		261		rollingpolicy.cpp		1	
265		262		rollingfileappender.cpp		1	
266		263		loggingevent.cpp		1	
267		264		gzcompressaction.cpp		1	
268		265		namepatternconverter.cpp		1	
269		266		outputdebugstringappender.cpp		0	
270		267		loggingeventpatternconverter.cpp		1	
271		268		simpledateformat.cpp		1	
272		269		filelocationpatternconverter.cpp		1	
273		270		threadspecificdata.cpp		1	

274		271		xmlsocketappender.cpp		1	
275		272		inputstream.cpp		1	
276		273		optionconverter.cpp		1	
277		274		appenderskeleton.cpp		1	
278		275		date.cpp		1	
279		276		locale.cpp		1	
280		277		propertysetter.cpp		1	
281		278		synchronized.cpp		1	
282		279		propertyresourcebundle.cpp		1	
283		280		bytebuffer.cpp		1	
284		281		literalpatternconverter.cpp		1	
285		282		simplelayout.cpp		1	
286		283		loggerpatternconverter.cpp		1	
287		284		levelrangefilter.cpp		1	
288		285		configurator.cpp		1	
289		286		ttcclayout.cpp		1	
290		287		syslogwriter.cpp		1	
291		288		manualtriggeringpolicy.cpp		1	
292		289		messagebuffer.cpp		1	
293		290		level.cpp		1	
294		291		threadcxx.cpp		1	
295		292		cyclicbuffer.cpp		1	
296		293		loglog.cpp		1	
297		294		fileappender.cpp		1	
298		295		mutex.cpp		1	
299		296		stringtokenizer.cpp		1	
300		297		filter.cpp		1	
301		298		datelayout.cpp		1	
302		299		action.cpp		1	
303		300		defaultloggerfactory.cpp		1	
304		301		loader.cpp		1	
305		302		domconfigurator.cpp		1	
306		303		levelpatternconverter.cpp		1	
307		304		relativetimedateformat.cpp		1	
308		305		serversocket.cpp		1	
309		306		timebasedrollingpolicy.cpp		1	
310		307		defaultconfigurator.cpp		1	
311		308		smtpappender.cpp		1	
312		309		rootlogger.cpp		1	
313		310		pool.cpp		1	
314		311		socketoutputstream.cpp		0	
315		312		messagepatternconverter.cpp		1	
316		313		transcoder.cpp		1	
317		314		dailyrollingfileappender.cpp		1	
318		315		defaultrepositoryselector.cpp		1	

```

319 | 316 | condition.cpp | 1 |
320 | 317 | lineseparatorpatternconverter.cpp | 1 |
321 | 318 | classnamepatternconverter.cpp | 1 |
322 | 319 | logger.cpp | 1 |
323 | 320 | mdc.cpp | 1 |
324 | 321 | xmllayout.cpp | 1 |
325 | 322 | ndcpatternconverter.cpp | 1 |
326 | 323 | bytearrayoutputstream.cpp | 1 |
327 | 324 | consoleappender.cpp | 1 |
328 | 325 | linelocationpatternconverter.cpp | 1 |
329 +-----+-----+-----+-----+
330 325 rows in set (10.57 sec)

```

and for the C++ Sockets Library the output is

The Individual File Weights for the C++ Sockets Library

```

1 +-----+-----+-----+-----+
2 | id | name | File Weight |
3 +-----+-----+-----+-----+
4 | 1 | Event.h | 4 |
5 | 2 | Debug.cpp | 0 |
6 | 3 | IHttpServer.h | 5 |
7 | 4 | SocketStream.cpp | 0 |
8 | 5 | HttpRequest.h | 6 |
9 | 6 | EventTime.h | 6 |
10 | 7 | HttpRequest.cpp | 2 |
11 | 8 | HttpPostSocket.h | 2 |
12 | 9 | Thread.cpp | 0 |
13 | 10 | HttpBaseSocket.h | 2 |
14 | 11 | MemFile.cpp | 0 |
15 | 12 | Ajp13Socket.cpp | 0 |
16 | 13 | HttpPostSocket.cpp | 2 |
17 | 14 | XmlException.cpp | 0 |
18 | 15 | SocketThread.cpp | 0 |
19 | 16 | EventHandler.h | 2 |
20 | 17 | Exception.h | 13 |
21 | 18 | IEventOwner.h | 4 |
22 | 19 | Lock.h | 9 |
23 | 20 | Sockets-config.cpp | 0 |
24 | 21 | Ipv4Address.h | 7 |
25 | 22 | ResolvServer.cpp | 0 |
26 | 23 | StreamSocket.cpp | 0 |
27 | 24 | SctpSocket.h | 0 |
28 | 25 | HttpResponse.cpp | 0 |
29 | 26 | EventHandler.cpp | 2 |

```

30		27		SocketStream.h		2	
31		28		SocketHandlerEp.cpp		0	
32		29		UdpSocket.cpp		0	
33		30		Mutex.cpp		0	
34		31		UdpSocket.h		4	
35		32		StdoutLog.cpp		0	
36		33		ResolvSocket.cpp		0	
37		34		StreamWriter.h		2	
38		35		HttpdCookies.cpp		0	
39		36		FileStream.h		2	
40		37		File.cpp		0	
41		38		Utility.cpp		0	
42		39		XmlDocument.h		0	
43		40		sockets-config.h		0	
44		41		EventTime.cpp		2	
45		42		HTTPSocket.cpp		0	
46		43		SctpSocket.cpp		0	
47		44		SocketThread.h		4	
48		45		socket_include.cpp		0	
49		46		Parse.cpp		0	
50		47		IFileUpload.h		3	
51		48		AjpBaseSocket.h		4	
52		49		Debug.h		2	
53		50		HttpdForm.h		6	
54		51		SocketHandlerThread.h		4	
55		52		SSLInitializer.cpp		0	
56		53		Ipv6Address.h		0	
57		54		SocketHandlerThread.cpp		2	
58		55		SmtpdSocket.cpp		0	
59		56		XmlNode.h		0	
60		57		Thread.h		50	
61		58		Base64.cpp		0	
62		59		SocketHandler.h		10	
63		60		Lock.cpp		2	
64		61		XmlException.h		0	
65		62		HttpTransaction.cpp		0	
66		63		FileStream.cpp		2	
67		64		Base64.h		4	
68		65		TcpSocket.cpp		0	
69		66		StreamWriter.cpp		2	
70		67		SocketHandler.cpp		2	
71		68		Json.h		2	
72		69		Mutex.h		40	
73		70		Exception.cpp		2	
74		71		IStream.h		9	

75		72		HttpBaseSocket.cpp		2	
76		73		HTTPSocket.h		16	
77		74		IFile.h		23	
78		75		ListenSocket.h		0	
79		76		ResolvSocket.h		4	
80		77		HttpResponse.h		6	
81		78		Semaphore.h		52	
82		79		IEventOwner.cpp		2	
83		80		IBase.h		1	
84		81		TcpSocket.h		30	
85		82		Semaphore.cpp		2	
86		83		XmlDocument.cpp		0	
87		84		HttpDebugSocket.h		2	
88		85		AjpBaseSocket.cpp		2	
89		86		SSLInitializer.h		32	
90		87		HttpDebugSocket.cpp		2	
91		88		socket_include.h		75	
92		89		SocketAddress.h		49	
93		90		StdLog.h		29	
94		91		HttpdSocket.h		2	
95		92		HttpPutSocket.h		2	
96		93		ISocketHandler.h		26	
97		94		HttpdCookies.h		10	
98		95		ResolvServer.h		4	
99		96		Ipv6Address.cpp		0	
100		97		XmlNode.cpp		0	
101		98		MemFile.h		6	
102		99		SmtpdSocket.h		2	
103		100		HttpGetSocket.h		2	
104		101		HttpClientSocket.h		8	
105		102		Ajp13Socket.h		2	
106		103		Event.cpp		2	
107		104		HttpPutSocket.cpp		2	
108		105		HttpdSocket.cpp		2	
109		106		HttpTransaction.h		10	
110		107		Ipv4Address.cpp		2	
111		108		IEventHandler.h		5	
112		109		Socket.h		46	
113		110		HttpClientSocket.cpp		2	
114		111		IMutex.h		43	
115		112		Parse.h		6	
116		113		Json.cpp		2	
117		114		Socket.cpp		2	
118		115		StreamSocket.h		35	
119		116		StdoutLog.h		2	


```

120 | 117 | HttpdForm.cpp          |          2 |
121 | 118 | File.h                 |         16 |
122 | 119 | ajp13.h                |          0 |
123 | 120 | Utility.h              |         47 |
124 | 121 | SocketHandlerEp.h     |          2 |
125 | 122 | HttpGetSocket.cpp     |          2 |
126 +-----+-----+-----+-----+
127 122 rows in set (0.34 sec)

```

A-2 Composite Query Output

A-2.1 Average SLOC for all Files

The output for the libodbc++ project for this query is

The Average SLOC for All Files in libodbc++

```

1 +-----+
2 | Average SLOC |
3 +-----+
4 |      391.5238 |
5 +-----+
6 1 row in set (0.11 sec)

```

for the log4cxx project the output is

The Average SLOC for All Files in log4cxx

```

1 +-----+
2 | Average SLOC |
3 +-----+
4 |      140.7169 |
5 +-----+
6 1 row in set (0.00 sec)

```

and for the C++ Sockets Library the output is

The Average SLOC for All Files in the C++ Sockets Library

```
1 +-----+
2 | Average SLOC |
3 +-----+
4 |      220.2377 |
5 +-----+
6 1 row in set (0.12 sec)
```

A-2.2 Average SLOC for Header Files

The output for the libodbc++ project for this query is

Average SLOC for Header Files in libodbc++

```
1 +-----+
2 | Average .h SLOC |
3 +-----+
4 |      278.6087 |
5 +-----+
6 1 row in set (0.04 sec)
```

for the log4cxx project the output is

Average SLOC for Header Files in log4cxx

```
1 +-----+
2 | Average .h SLOC |
3 +-----+
4 |      126.3051 |
5 +-----+
6 1 row in set (0.00 sec)
```

and for the C++ Sockets Library the output is

Average SLOC for Header Files in the C++ Sockets Library

```
1 +-----+
2 | Average .h SLOC |
3 +-----+
4 |      124.0299 |
5 +-----+
6 1 row in set (0.02 sec)
```

A-2.3 Average SLOC for Source Files

The output for the libodbc++ project for this query is

Average SLOC for Source Files in libodbc++

```
1 +-----+
2 | Average .cpp SLOC |
3 +-----+
4 |           528.2105 |
5 +-----+
6 1 row in set (0.07 sec)
```

for the log4cxx project the output is

Average SLOC for Source Files in log4cxx

```
1 +-----+
2 | Average .cpp SLOC |
3 +-----+
4 |           157.9527 |
5 +-----+
6 1 row in set (0.00 sec)
```

and for the C++ Sockets Library the output is

Average SLOC for Source Files in the C++ Sockets Library

```
1 +-----+
2 | Average .cpp SLOC |
3 +-----+
4 |           337.4364 |
5 +-----+
6 1 row in set (0.09 sec)
```

A-2.4 Proportion of Private, Protected and Public Variables

The output for the libodbc++ project for this query is

Proportion of Variables for libodbc++

```
1 +-----+
2 | Private to Protected to Public |
3 +-----+
4 | 81 to 0 to 311 |
5 +-----+
6 1 row in set (0.09 sec)
```

for the log4cxx project the output is

Proportion of Variables for log4cxx

```
1 +-----+
2 | Private to Protected to Public |
3 +-----+
4 | 23 to 0 to 1029 |
5 +-----+
6 1 row in set (0.92 sec)
```

and for the C++ Sockets Library the output is

Proportion of Variables for the C++ Sockets Library

```
1 +-----+
2 | Private to Protected to Public |
3 +-----+
4 | 40 to 1 to 244 |
5 +-----+
6 1 row in set (0.05 sec)
```

A-2.5 Percentage Private Variables

The output for the libodbc++ project for this query is

Percentage Private Variables for libodbc++

```
1 +-----+
2 | Percentage Private Variables |
3 +-----+
4 | 20.6633% |
5 +-----+
6 1 row in set (0.08 sec)
```

for the log4cxx project the output is

Percentage Private Variables for log4cxx

```
1 +-----+
2 | Percentage Private Variables |
3 +-----+
4 | 2.1905%                      |
5 +-----+
6 1 row in set (0.81 sec)
```

and for the C++ Sockets Library the output is

Percentage Private Variables for the C++ Sockets Library

```
1 +-----+
2 | Percentage Private Variables |
3 +-----+
4 | 14.0351%                     |
5 +-----+
6 1 row in set (0.05 sec)
```

A-2.6 Percentage Protected Variables

The output for the libodbc++ project for this query is

Percentage Protected Variables for libodbc++

```
1 +-----+
2 | Percentage Protected Variables |
3 +-----+
4 | 0.0000%                       |
5 +-----+
6 1 row in set (0.08 sec)
```

for the log4cxx project the output is

Percentage Protected Variables for log4cxx

```
1 +-----+
2 | Percentage Protected Variables |
3 +-----+
4 | 0.0000%                       |
5 +-----+
6 1 row in set (0.82 sec)
```

and for the C++ Sockets Library the output is

Percentage Protected Variables for the C++ Sockets Library

```
1 +-----+
2 | Percentage Protected Variables |
3 +-----+
4 | 0.3509%                        |
5 +-----+
6 1 row in set (0.11 sec)
```

A-2.7 Percentage Public Variables

The output for the libodbc++ project for this query is

Percentage Public Variables for libodbc++

```
1 +-----+
2 | Percentage Public Variables |
3 +-----+
4 | 79.3367%                    |
5 +-----+
6 1 row in set (0.13 sec)
```

for the log4cxx project the output is

Percentage Public Variables for log4cxx

```
1 +-----+
2 | Percentage Public Variables |
3 +-----+
4 | 98.0000%                     |
5 +-----+
6 1 row in set (1.14 sec)
```

and for the C++ Sockets Library the output is

Percentage Public Variables for the C++ Sockets Library

```
1 +-----+
2 | Percentage Public Variables |
3 +-----+
4 | 85.6140%                      |
5 +-----+
6 1 row in set (0.18 sec)
```

A-2.8 Set of Queries to Calculate Percentage of Files Referenced

The output for the libodbc++ project for this query is

The Percent Code Used in libodbc++

```
1 +-----+
2 | File_Weight |
3 +-----+
4 |           1 |
5 |           7 |
6 |           1 |
7 |           1 |
8 |           1 |
9 |           1 |
10 |          10 |
11 |           1 |
12 |           4 |
13 |           1 |
14 |           1 |
15 |           2 |
16 |           1 |
17 |           2 |
18 |          28 |
19 |           6 |
20 |          14 |
21 |           4 |
22 |           7 |
23 |           3 |
24 |          21 |
25 |          19 |
26 |           7 |
27 |           9 |
28 |           2 |
29 |           2 |
30 |           2 |
31 |           1 |
32 |           3 |
33 |           2 |
34 +-----+
35 30 rows in set (0.19 sec)
36
37 +-----+
38 | Number Files With Dependencies |
```

```

39 +-----+
40 |                               30 |
41 +-----+
42 1 row in set (0.00 sec)
43
44 +-----+
45 | Percentage of Files Referenced |
46 +-----+
47 | 71.4286%                       |
48 +-----+
49 1 row in set (0.00 sec)

```

for the log4cxx project the output is

The Percent Code Used in log4cxx

```

1 +-----+
2 | File_Weight |
3 +-----+
4 |           3 |
5 |           3 |
6 |           3 |
7 |           9 |
8 |          56 |
9 |           3 |
10 |           3 |
11 |           2 |
12 |          40 |
13 |           4 |
14 |           4 |
15 |           4 |
16 |           3 |
17 |           1 |
18 |           1 |
19 |           1 |
20 |           3 |
21 |           3 |
22 |           1 |
23 |           2 |
24 |           1 |
25 |          174 |
26 |           4 |
27 |          175 |
28 |           3 |
29 |          178 |
30 |           6 |

```


31		20	
32		176	
33		176	
34		45	
35		57	
36		174	
37		13	
38		134	
39		5	
40		4	
41		7	
42		186	
43		54	
44		26	
45		5	
46		26	
47		27	
48		3	
49		175	
50		1	
51		7	
52		5	
53		23	
54		272	
55		3	
56		276	
57		2	
58		178	
59		4	
60		13	
61		4	
62		3	
63		5	
64		7	
65		11	
66		1	
67		28	
68		3	
69		272	
70		11	
71		2	
72		3	
73		3	
74		3	
75		8	

76		4	
77		193	
78		4	
79		6	
80		19	
81		6	
82		4	
83		20	
84		16	
85		9	
86		20	
87		4	
88		262	
89		4	
90		41	
91		270	
92		4	
93		306	
94		9	
95		6	
96		3	
97		3	
98		5	
99		2	
100		176	
101		5	
102		183	
103		17	
104		3	
105		100	
106		178	
107		221	
108		3	
109		29	
110		317	
111		19	
112		5	
113		21	
114		4	
115		10	
116		29	
117		4	
118		21	
119		9	
120		4	

121		4	
122		4	
123		4	
124		4	
125		5	
126		3	
127		16	
128		3	
129		4	
130		3	
131		18	
132		3	
133		44	
134		3	
135		3	
136		3	
137		3	
138		3	
139		5	
140		4	
141		7	
142		9	
143		3	
144		3	
145		3	
146		58	
147		3	
148		5	
149		3	
150		3	
151		3	
152		16	
153		177	
154		11	
155		4	
156		7	
157		1	
158		1	
159		16	
160		3	
161		3	
162		3	
163		7	
164		3	
165		4	

166		3	
167		306	
168		3	
169		176	
170		2	
171		39	
172		184	
173		3	
174		4	
175		2	
176		174	
177		5	
178		1	
179		1	
180		1	
181		1	
182		1	
183		1	
184		1	
185		1	
186		1	
187		1	
188		1	
189		1	
190		1	
191		1	
192		1	
193		1	
194		1	
195		1	
196		1	
197		1	
198		1	
199		1	
200		1	
201		1	
202		1	
203		1	
204		1	
205		1	
206		1	
207		1	
208		1	
209		1	
210		1	

211		1	
212		1	
213		1	
214		1	
215		1	
216		1	
217		1	
218		1	
219		1	
220		1	
221		1	
222		1	
223		1	
224		1	
225		1	
226		1	
227		1	
228		1	
229		1	
230		1	
231		1	
232		1	
233		1	
234		1	
235		1	
236		1	
237		1	
238		1	
239		1	
240		1	
241		1	
242		1	
243		1	
244		1	
245		1	
246		1	
247		1	
248		1	
249		1	
250		1	
251		1	
252		1	
253		1	
254		1	
255		1	

256		1	
257		1	
258		1	
259		1	
260		1	
261		1	
262		1	
263		1	
264		1	
265		1	
266		1	
267		1	
268		1	
269		1	
270		1	
271		1	
272		1	
273		1	
274		1	
275		1	
276		1	
277		1	
278		1	
279		1	
280		1	
281		1	
282		1	
283		1	
284		1	
285		1	
286		1	
287		1	
288		1	
289		1	
290		1	
291		1	
292		1	
293		1	
294		1	
295		1	
296		1	
297		1	
298		1	
299		1	
300		1	

```

301 |          1 |
302 |          1 |
303 |          1 |
304 |          1 |
305 |          1 |
306 |          1 |
307 |          1 |
308 |          1 |
309 |          1 |
310 |          1 |
311 |          1 |
312 |          1 |
313 |          1 |
314 |          1 |
315 |          1 |
316 |          1 |
317 |          1 |
318 |          1 |
319 |          1 |
320 |          1 |
321 |          1 |
322 +-----+
323 318 rows in set (0.00 sec)
324
325 +-----+
326 | Number Files With Dependencies |
327 +-----+
328 |                               318 |
329 +-----+
330 1 row in set (0.00 sec)
331
332
333 +-----+
334 | Percentage of Files Referenced |
335 +-----+
336 | 97.8462%                       |
337 +-----+
338 1 row in set (0.00 sec)

```

and for the C++ Sockets Library the output is

The Percent Code Used in the C++ Sockets Library

```

1 +-----+
2 | File_Weight |
3 +-----+

```

4		4	
5		5	
6		6	
7		6	
8		2	
9		2	
10		2	
11		2	
12		2	
13		13	
14		4	
15		9	
16		7	
17		2	
18		2	
19		4	
20		2	
21		2	
22		2	
23		4	
24		3	
25		4	
26		2	
27		6	
28		4	
29		2	
30		50	
31		10	
32		2	
33		2	
34		4	
35		2	
36		2	
37		2	
38		40	
39		2	
40		9	
41		2	
42		16	
43		23	
44		4	
45		6	
46		52	
47		2	
48		1	


```

49 |          30 |
50 |           2 |
51 |           2 |
52 |           2 |
53 |          32 |
54 |           2 |
55 |          75 |
56 |          49 |
57 |          29 |
58 |           2 |
59 |           2 |
60 |          26 |
61 |          10 |
62 |           4 |
63 |           6 |
64 |           2 |
65 |           2 |
66 |           8 |
67 |           2 |
68 |           2 |
69 |           2 |
70 |           2 |
71 |          10 |
72 |           2 |
73 |           5 |
74 |          46 |
75 |           2 |
76 |          43 |
77 |           6 |
78 |           2 |
79 |           2 |
80 |          35 |
81 |           2 |
82 |           2 |
83 |          16 |
84 |          47 |
85 |           2 |
86 |           2 |
87 +-----+
88 83 rows in set (1.65 sec)
89
90 +-----+
91 | Number Files With Dependencies |
92 +-----+
93 |                               83 |

```

```

94 +-----+
95 1 row in set (0.00 sec)
96
97 +-----+
98 | Percentage of Files Referenced |
99 +-----+
100 | 68.0327868852459%           |
101 +-----+
102 1 row in set (0.00 sec)

```

A-2.9 Percent Code Reused

The output for the libodbc++ project for this query is

Percent Code Reused by libodbc++

```

1 +-----+
2 | Percent Code Reused |
3 +-----+
4 | 404.7619%           |
5 +-----+
6 1 row in set (0.93 sec)

```

for the log4cxx project the output is

Percent Code Reused by log4cxx

```

1 +-----+
2 | Percent Code Reused |
3 +-----+
4 | 48.9231%           |
5 +-----+
6 1 row in set (1 min 3.72 sec)

```

and for the C++ Sockets Library the output is

Percent Code Reused by the C++ Sockets Library

```

1 +-----+
2 | Percent Code Reused |
3 +-----+
4 | 1.1393%            |
5 +-----+
6 1 row in set (8.56 sec)

```

A-3 Advanced Query Output

A-3.1 Search Source for a String

Search Source for 'Event::Data() const'

1	+-----+-----+-----+-----+			
2	name	id	file_id	located_at_char
3	+-----+-----+-----+-----+			
4	Event.h	1	1	0
5	Debug.cpp	2	2	0
6	IHttpServer.h	3	3	0
7	SocketStream.cpp	4	4	0
8	HttpRequest.h	5	5	0
9	EventTime.h	6	6	0
10	HttpRequest.cpp	7	7	0
11	HttpPostSocket.h	8	8	0
12	Thread.cpp	9	9	0
13	HttpBaseSocket.h	10	10	0
14	MemFile.cpp	11	11	0
15	Ajp13Socket.cpp	12	12	0
16	HttpPostSocket.cpp	13	13	0
17	XmlException.cpp	14	14	0
18	SocketThread.cpp	15	15	0
19	EventHandler.h	16	16	0
20	Exception.h	17	17	0
21	IEventOwner.h	18	18	0
22	Lock.h	19	19	0
23	Sockets-config.cpp	20	20	0
24	Ipv4Address.h	21	21	0
25	ResolveServer.cpp	22	22	0
26	StreamSocket.cpp	23	23	0
27	SctpSocket.h	24	24	0
28	HttpResponse.cpp	25	25	0
29	EventHandler.cpp	26	26	0
30	SocketStream.h	27	27	0
31	SocketHandlerEp.cpp	28	28	0
32	UdpSocket.cpp	29	29	0
33	Mutex.cpp	30	30	0
34	UdpSocket.h	31	31	0
35	StdoutLog.cpp	32	32	0
36	ResolveSocket.cpp	33	33	0
37	StreamWriter.h	34	34	0
38	HttpdCookies.cpp	35	35	0
39	FileStream.h	36	36	0

40	File.cpp	37	37	0
41	Utility.cpp	38	38	0
42	XmlDocument.h	39	39	0
43	sockets-config.h	40	40	0
44	EventTime.cpp	41	41	0
45	HTTPSocket.cpp	42	42	0
46	SctpSocket.cpp	43	43	0
47	SocketThread.h	44	44	0
48	socket_include.cpp	45	45	0
49	Parse.cpp	46	46	0
50	IFileUpload.h	47	47	0
51	AjpBaseSocket.h	48	48	0
52	Debug.h	49	49	0
53	HttpdForm.h	50	50	0
54	SocketHandlerThread.h	51	51	0
55	SSLInitializer.cpp	52	52	0
56	Ipv6Address.h	53	53	0
57	SocketHandlerThread.cpp	54	54	0
58	SmtpdSocket.cpp	55	55	0
59	XmlNode.h	56	56	0
60	Thread.h	57	57	0
61	Base64.cpp	58	58	0
62	SocketHandler.h	59	59	0
63	Lock.cpp	60	60	0
64	XmlException.h	61	61	0
65	HttpTransaction.cpp	62	62	0
66	FileStream.cpp	63	63	0
67	Base64.h	64	64	0
68	TcpSocket.cpp	65	65	0
69	StreamWriter.cpp	66	66	0
70	SocketHandler.cpp	67	67	0
71	Json.h	68	68	0
72	Mutex.h	69	69	0
73	Exception.cpp	70	70	0
74	IStream.h	71	71	0
75	HttpBaseSocket.cpp	72	72	0
76	HTTPSocket.h	73	73	0
77	IFile.h	74	74	0
78	ListenSocket.h	75	75	0
79	ResolvSocket.h	76	76	0
80	HttpResponse.h	77	77	0
81	Semaphore.h	78	78	0
82	IEventOwner.cpp	79	79	0
83	IBase.h	80	80	0
84	TcpSocket.h	81	81	0

85	Semaphore.cpp	82	82	0	
86	XmlDocument.cpp	83	83	0	
87	HttpDebugSocket.h	84	84	0	
88	AjpBaseSocket.cpp	85	85	0	
89	SSLInitializer.h	86	86	0	
90	HttpDebugSocket.cpp	87	87	0	
91	socket_include.h	88	88	0	
92	SocketAddress.h	89	89	0	
93	StdLog.h	90	90	0	
94	HttpdSocket.h	91	91	0	
95	HttpPutSocket.h	92	92	0	
96	ISocketHandler.h	93	93	0	
97	HttpdCookies.h	94	94	0	
98	ResolveServer.h	95	95	0	
99	Ipv6Address.cpp	96	96	0	
100	XmlNode.cpp	97	97	0	
101	MemFile.h	98	98	0	
102	SmtpdSocket.h	99	99	0	
103	HttpGetSocket.h	100	100	0	
104	HttpClientSocket.h	101	101	0	
105	Ajp13Socket.h	102	102	0	
106	Event.cpp	103	103	1860	
107	HttpPutSocket.cpp	104	104	0	
108	HttpdSocket.cpp	105	105	0	
109	HttpTransaction.h	106	106	0	
110	Ipv4Address.cpp	107	107	0	
111	IEventHandler.h	108	108	0	
112	Socket.h	109	109	0	
113	HttpClientSocket.cpp	110	110	0	
114	IMutex.h	111	111	0	
115	Parse.h	112	112	0	
116	Json.cpp	113	113	0	
117	Socket.cpp	114	114	0	
118	StreamSocket.h	115	115	0	
119	StdoutLog.h	116	116	0	
120	HttpdForm.cpp	117	117	0	
121	File.h	118	118	0	
122	ajp13.h	119	119	0	
123	Utility.h	120	120	0	
124	SocketHandlerEp.h	121	121	0	
125	HttpGetSocket.cpp	122	122	0	
126	+-----+-----+-----+-----+-----+				
127	122 rows in set (0.06 sec)				

A-3.2 Search XML for a String

Search XML for 'Event::Data() const'

	name	id	file_id	located_at_char
1				
2	name	id	file_id	located_at_char
3				
4	Event.xml	1	123	63660
5	Debug.xml	2	124	0
6	IHttpServer.xml	3	125	0
7	SocketStream.xml	4	126	0
8	HttpRequest.xml	5	127	0
9	EventTime.xml	6	128	0
10	HttpPostSocket.xml	7	129	0
11	Thread.xml	8	130	0
12	HttpBaseSocket.xml	9	131	0
13	MemFile.xml	10	132	0
14	Ajp13Socket.xml	11	133	0
15	XmlException.xml	12	134	0
16	SocketThread.xml	13	135	0
17	EventHandler.xml	14	136	0
18	Exception.xml	15	137	0
19	IEventOwner.xml	16	138	0
20	Lock.xml	17	139	0
21	Ipv4Address.xml	18	141	0
22	ResolveServer.xml	19	142	0
23	StreamSocket.xml	20	143	0
24	SctpSocket.xml	21	144	0
25	HttpResponse.xml	22	145	0
26	SocketHandlerEp.xml	23	146	0
27	UdpSocket.xml	24	147	0
28	Mutex.xml	25	148	0
29	StdoutLog.xml	26	149	0
30	ResolveSocket.xml	27	150	0
31	StreamWriter.xml	28	151	0
32	HttpdCookies.xml	29	152	0
33	FileStream.xml	30	153	0
34	File.xml	31	154	0
35	Utility.xml	32	155	0
36	XmlDocument.xml	33	156	0
37	HTTPSocket.xml	34	157	0
38	socket_include.xml	35	158	0
39	Parse.xml	36	159	0
40	IFileUpload.xml	37	160	0
41	AjpBaseSocket.xml	38	161	0
42	HttpdForm.xml	39	162	0

```

43 | SocketHandlerThread.xml | 40 | 163 | 0 |
44 | SSLInitializer.xml | 41 | 164 | 0 |
45 | Ipv6Address.xml | 42 | 165 | 0 |
46 | SmtpdSocket.xml | 43 | 166 | 0 |
47 | XmlNode.xml | 44 | 167 | 0 |
48 | Base64.xml | 45 | 168 | 0 |
49 | SocketHandler.xml | 46 | 169 | 0 |
50 | HttpTransaction.xml | 47 | 170 | 0 |
51 | TcpSocket.xml | 48 | 171 | 0 |
52 | Json.xml | 49 | 172 | 0 |
53 | IStream.xml | 50 | 173 | 0 |
54 | IFile.xml | 51 | 174 | 0 |
55 | ListenSocket.xml | 52 | 175 | 0 |
56 | Semaphore.xml | 53 | 176 | 0 |
57 | IBase.xml | 54 | 177 | 0 |
58 | HttpDebugSocket.xml | 55 | 178 | 0 |
59 | SocketAddress.xml | 56 | 179 | 0 |
60 | StdLog.xml | 57 | 180 | 0 |
61 | HttpdSocket.xml | 58 | 181 | 0 |
62 | HttpPutSocket.xml | 59 | 182 | 0 |
63 | ISocketHandler.xml | 60 | 183 | 0 |
64 | HttpGetSocket.xml | 61 | 184 | 0 |
65 | HttpClientSocket.xml | 62 | 185 | 0 |
66 | IEventHandler.xml | 63 | 186 | 0 |
67 | Socket.xml | 64 | 187 | 0 |
68 | IMutex.xml | 65 | 188 | 0 |
69 | ajp13.xml | 66 | 189 | 0 |
70 +-----+-----+-----+-----+
71 66 rows in set (8.36 sec)

```

A-3.3 Find Code Surrounding a String in Source

Find Code Surrounding 'Event::Data() const' in Source

```

1 +-----+-----+-----+-----+
2 | name      | id | file_id | data          |
3 +-----+-----+-----+-----+
4 | Event.h | 1 | 1 | / _SOCKETS_Event_H
5
6 |
7 +-----+-----+-----+-----+
8 1 row in set (0.00 sec)

```

A-3.4 Find Code Surrounding a String in XML

Find Code Surrounding 'Event::Data() const' in XML

```
1 +-----+-----+-----+-----+
2 | name      | id | file_id | data                                     |
3 +-----+-----+-----+-----+
4 | Event.xml | 1  |      123 | 4DataEv" demangled="Event::Data() const" |
5 +-----+-----+-----+-----+
6 1 row in set (0.01 sec)
```