REportal: A Web-based Portal Site for Reverse Engineering

Spiros Mancoridis, Timothy S. Souder
Department of Mathematics & Computer Science
Drexel University
Philadelphia, PA, USA
{smancori,gtsouder}@mcs.drexel.edu

Yih-Farn Chen, Emden R. Gansner, Jeffrey L. Korn
AT&T Labs - Research
Florham Park, NJ 07932, USA
{chen,erg,jlk}@research.att.com

Abstract

We present a web-based portal site for reverse engineering software systems called REportal. REportal *enables authorized users to upload their code to a secure web site and then, through the guide of wizards, browse and analyze their code. Currently the portal services include code analysis, browsing, querying, and design extraction for C, C++, and Java programs. The* REportal *services are implemented by several reverse engineering tools that our team has developed over the years. With this work, we aim to assist professional software engineers, educators, and other researchers who need to analyze code. Specifically, we present a technology that provides a simple and easily accessible user interface to a number of reverse engineering tools. More importantly, this technology saves the user from the time and effort required to install, administer, and integrate these tools.*

1. Introduction

Over the years the Reverse Engineering research community has produced a plethora of tools to support tasks such as source code analysis, program browsing, design extraction, dynamic analysis, and so on. In many cases, these tools are developed to demonstrate the effectiveness of research ideas and techniques. Most often, the usability, interoperability, reliability, and ease of installation of these tools is, at best, considered as an afterthought. As a result, professional software engineers, educators, and other researchers frequently decide not to use these tools.

Our team has developed several reverse engineering tools such as: the Acacia and Chava source code analyzers for C, C++, and Java respectively, the CIAO [11] source code browser, the Bunch software clustering tool, and the Form framework for profiling C, C++, and Java programs. Although we spent considerable time designing these tools with end-users in mind, our experience has been that using these tools on production software has a significant learning curve.

Our solution to this problem is a web-based portal site for reverse engineering software systems called REportal. REportal enables authorized users to upload their code to a secure web site and then, through the guide of wizards, browse and analyze their code. Currently the portal services include code analysis, browsing, querying, and design extraction for C, C++, and Java programs.

Figure 1 shows one interface where users can choose from an array of services. These services create views that show class hierarchies, call graphs, source-level information, and include hypertext-based source code browsing and searching.

The rest of this paper is structured as follows: in Section 2, we provide an overview of related work, in Section 3, we describe the reverse engineering services currently supported by REportal, in Section 4, we outline the software architecture of REportal, in Section 5 we mention our plans to improve our site and, finally, in Section 6, we conclude by summarizing the paper and outlining the contributions of our work.

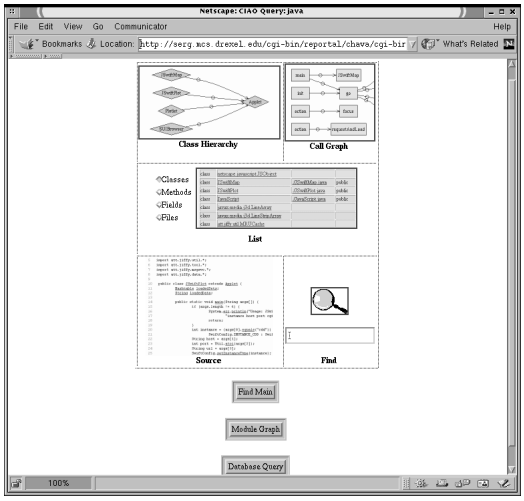

Figure 1. REportal Service

2. Background

The REportal project is closely related to work in the areas of web-based software engineering, code analysis, and design extraction.

2.1. Web-based Software Engineering

The *Reverse and Reengineering Roadmap* [3] is a portal site for reverse engineering established and maintained by van Deursen. Unlike REportal, it does not offer code analysis and browsing services but is rather a hub to information about reverse engineering, program understanding, and software evolution research and practice.

There are several other examples of using the Web for software engineering. Holt et al [15] have developed the *Software Bookshelf*, which is a web-based paradigm for the presentation and navigation of information representing large software systems. Holt has also set up a web site at the University of Waterloo with the purpose of making a number of "guinea pig" [20] software systems, along with analyses performed on these systems, publically available to the reverse engineering community. Some of the "guinea pigs" on the web site are Netscape/Mozilla, Apache, GNU CC, VIM, and the Linux Kernel. Muller, at the University of Victoria, has a similar web site [19].

The Source Forge web site [36] is an example of how the Web can be used to facilitate software development, maintenance, and distribution. Sourceforge is a free service to Open Source developers offering access to CVS repositories, mailing lists, bug tracking, message boards/forums, task management, site hosting, permanent file archival, full backups, and web-based administration. Similarly, the Freshmeat web site [17] is a software distribution site from which a variety of Open Source software may be accessed.

REportal is an example of "Software as Services" that is becoming very popular in the IT industry, as exemplified by initiatives from Microsoft's .Net/HailStorm [26] and Sun Microsystems' Web Services [38].

As many software engineering tools operate on graphs or produce graphs as output, efforts in the graph drawing community to make tools accessible via the Web are relevant to this work. There are a few examples of servers for drawing graphs. The first such service was *drawdag* at AT&T. The user sends a dot file to `drawdag@research.att.com`, with the layout format and options specified on the `Subject:` line. The server does the layout, and then mails back the output. Currently active is the *WebDot* [40] server at AT&T. This allows web pages to contain image anchors such that, when the page is being read by the browser, a dot file is read by the WebDot server, the graph is laid out, and the output is fed back to the browser for inclusion in the page. A similar web-based graph drawing server exists at Brown University [7, 8].

2.2. Code Analysis

There has been considerable progress in code analysis tools for C, C++, Java, and COBOL. Code analysis tools parse source or intermediate (e.g., bytecode) code and produce a database of code entities (e.g., functions and variables) and relationships (e.g., method invocation and object instantiation). When trying to understand the code of a program, developers formulate queries against the code database such as: (a) What is the inheritance hierarchy rooted at class C? (b) Is function f_1 reachable from function f_2 through a sequence of calls? (c) What are the signatures of the constructors of class C? The results of such queries are typically displayed as simple text or as a graph using visualization tools such as dot [18, 30].

Two code analyzers have been developed at AT&T: Acacia [10] for C/C++ and Chava [22] for Java (both tools are available on-line [4]). These tools are described in more detail in Section 3. Other examples of code analyzers are Reprise [32] for C++ and Cobol/SRE [29] for COBOL. In addition to code analysis tools, there has been work in meta-tool code analysis generators such as Aria [13] and Genoa [12].

Early research on code analysis has influenced the more current advances in commercial tools.

Red Hat's Source-Navigator [37] is a code analysis tool that supports C, C++, Java, Tcl, FORTRAN and COBOL. Using this tool one can display relationships between classes, functions, and members, find every place in the code where a given function is called, find each file that includes a given header file, display call trees, and so on.

Netcomputing's AnyJ [2] is a Java IDE and code analysis tool. In addition to the typical tools offered by an IDE (e.g., editor, debugger), AnyJ includes code browsing and analysis tools that operate on .zip, .class, .jar, or .java files. For example, AnyJ allows programmers to jump to the definition of methods/variables with respect to the type of the current expression and class hierarchy.

Western Ware's CC-Rider [9] is a code analysis and browsing tool to visualize and document software written in C and C++. CC-Rider's visual browsing supports a variety of Tree Views such as: class hierarchy, class ancestry, class nesting, function call, data usage, and file usage. Similar to REportal, CC-Rider supports code navigation using Hypertext links.

2.3. Design Extraction

Early work by Belady and Evangelist [6] identified automatic design extraction (also known as software clustering [41]) as a means to produce high-level views of the structure of a software system from its code. Much of the earlier work in this area, like that of Hutchens and Basili [21], focuses on techniques for recovering code-level (concrete) designs by grouping related procedures and variables into modules. Progressively, as software systems grew in size, the new problem of recovering design-level (abstract) designs by grouping sets of modules into hierarchies of subsystems became pertinent. Schwanke's ARCH tool [34] introduced concepts such as low-coupling and high-cohesion into the design recovery problem. The Rigi system [27] by Muller et al, pioneered the concepts of isolating omnipresent modules, grouping modules with common clients and suppliers, and grouping modules that have similar names. The last idea was followed-up by Anquetil and Lethbridge [1], who used common patterns in file names as a clustering criterion. Our Bunch tool [24, 23] treats design recovery as an optimization problem and was the first system to employ genetic algorithms [14] to overcome the problem of getting "stuck" on local optima solutions. Recently, Tzerpos and Holt [39] created a clustering algorithm that uses graph theory algorithms in conjunction with an elaborate multi-phase process. Finally, Murphy et al [28] take a top-down approach to design recovery using tools to capture differences between the actual source code organization and the designer's mental model of the high-level software design.

3. REportal Services

3.1. Code Analysis Service

At the core of the reverse engineering services provided by REportal is a database that captures a software engineering view of a program or system. The schema for this database supports a representation of programs as a collection of software entities, such as files, functions, types and variables, and relations among pairs of entities, such as "function a" calls "function b", or "class D" is a subclass of "class C". At present, REportal is focused on the analysis of C/C++ and Java code. It provides two analysis engines, one for each language family, that can analyze a program and generate a database with the appropriate information, and in the supported format.

As REportal relies on the CIAO [11] architecture, the ability for REportal to handle a new language requires little more than devising an appropriate database schema and writing an analysis tool to create the database.

3.1.1. C/C++ Code Analysis Service

By default, REportal uses Acacia for the analysis of C and C++ programs. Though more fully described elsewhere [10], we give a brief description of its features and structure. Acacia mimics the typical C compiler interface, in that one can build a database by replacing every use of the C or C++ compiler by a call to Acacia. In particular, it accepts the conventional compiler flags, and is typically used to process a collection of source files, followed by a final pass that links the individual databases into a single program database, analogous to the linking of a collection of object files into a single executable program.

The analysis accepts source code written in ISO C/C++ and K&R C. An initial preprocessor pass is performed, allowing Acacia to catch macro definitions and uses. This is important, as the database is meant to give a source level view of the program. Acacia then performs parsing and type checking, which results in a annotated parse tree. From the parse tree and related tables, it extracts the entities and relations needed by REportal.

One of the difficulties in the analysis of C or C++ is the non-portability inherent in any significant program, despite international standardization. This is in

distinction to Java code, which, at present, is portable, at least at the level of compilation. The problems in the C family tend to arise from three causes.

- There are many flavors of C and C++. Despite the existence of standards, some compiler providers have yet to implement them, sometimes due to the complexity of the standards themselves. In other cases, compiler providers have intentionally introduced new, non-standard C or C++ syntax. These additions to the syntax occur in such popular compilers as *gcc* and *Visual C++*. In either case, a programmer will often use the constructs without even realizing they are not standards-compliant. In a bow to the popularity of *gcc*, Acacia includes various fixes to finesse some of the more frequent non-standard constructs.

- In the C family, much of the system level functionality is not standardized, especially as regards implementation and the use of include files. This means that it is easy to write code that compiles on one system but not on another. At best, even if the programmer makes the code portable by the disciplined use of compiler directives (e.g., `#ifdef`, etc.), the resulting databases will vary greatly from system to system, especially concerning the low-level, system entities.

- The build process typically relies on some version of the Unix *make* tool, which in turn often relies on various other programs used to configure the build, or generate code and data. This further magnifies the portability problem, and can make the automatic generation of a REportal database difficult.

There are various approaches to resolve these problems. Principally, if the analysis is to be done by REportal, we require that the user code be written in one of the three supported dialects; that the code be written portably so that it will build using the system include files on the REportal server; and that the user provides a makefile script that does not need to build any intermediate tools. If it is important that the analysis captures the low-level structure of a specific system, the user can supply, as part of the program source, a collection of include files from the appropriate system. Alternatively, as Acacia is available independently of REportal, the user can perform the analysis locally, and then copy the databases to the server.

We hope that, as REportal develops, it will be able to support multiple platforms more cleanly, either by providing the necessary include files and Acacia databases of system libraries, or by including different platforms as part of the service.

3.1.2. Java Code Analysis Service

REportal uses Chava [22] for the analysis of Java programs. Chava extracts information from Java applets or applications about classes, methods, fields and their relationships. This information is then stored in a CIAO relational database. Given our Java data model, we can query, visualize, and analyze the structural information with the language independent CIAO tools.

Chava is able to process either Java source files or compiled class files, making it possible to analyze remote applets whose source code is unavailable. Analysis using only class files is possible primarily due to properties of the Java language. Java's absence of a preprocessor means that we do not have to deal with constructs such as macros, include files, and templates, whose information would not be available in an object file. Also, Java is an architecture-neutral language, so it is possible to scan through object code in a machine-independent manner to discover relationships in a program.

Performance numbers indicate that the tool scales well. Running times indicate that Chava runs an order of magnitude faster than `javac` (the Java compiler). In fact, Chava is also faster than `javap`, the Java program that dumps the contents of a class file. The generated database size is in the order of the size of the class files, which is quite manageable. Size could be reduced significantly if compression is used on the database. The number of entities and relationships is small enough that queries can be performed efficiently.

3.2. Code Querying and Browsing Service

3.2.1. Code Querying Service

REportal's database query window (Figure 2) allows power users to perform customized entity or relationship queries that explore various structural aspects of the C, C++, or Java program being analyzed.

An entity query allows a user to select database records based on attribute values specified only on the left column in the query table. A relationship query examines relationships among entities by specifying attribute values of the source entity (left column) and the sink entity (right column) of a relationship.

For example, the query specified in Figure 2 asks the following question:

"Which public methods (functions) under the class `MapData` *read from fields (variables) in the class* `USAMap`*?"*

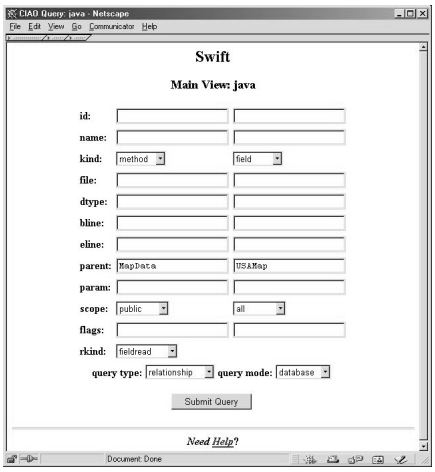

Figure 2. Source Code Querying

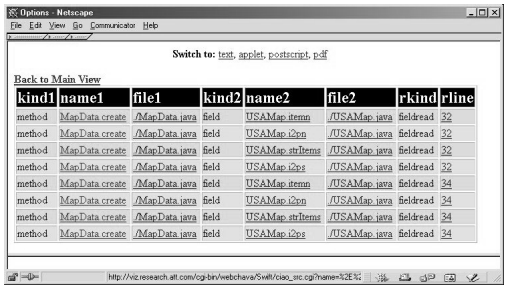

Figure 3. Source Code Querying Results (Database Mode)

The result can be presented in database mode (list of records as shown in Figure 3), graph mode (laying out the relationships as a graph), or text mode (show the source lines where those references occur). Presentation is accomplished by piping the intermediate database to the corresponding presentation tool.

3.2.2. Source Code Browsing

REportal can display source code as hyper-linked source in a web browser, as shown in Figure 4. When source code refers to a program entity, a link is placed in the resulting HTML output. If a user follows a link, he can see where the entity is defined, or perform a set of queries on that entity.

A file's source code is transformed into HTML using a novel approach. A very simple lexical analyzer tokenizes the source to find identifiers. Then, CIAO consults the entity-relationship database of the program to find the set of the relationships originating from the

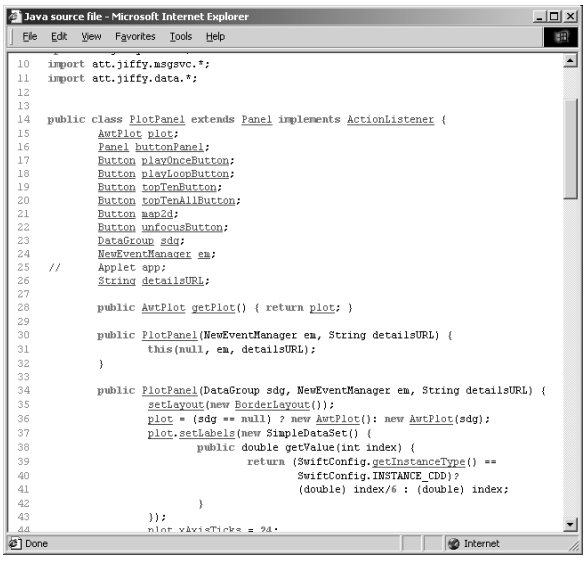

Figure 4. Source Code Browsing

file. When identifiers appear on the same line where a relationship exists, and the relationship is between an entity with the same name as that identifier, it is rendered as a hyper-links. In instances where there are different entities with the same name on the same line, this approach will fail. However, in practice this does not happen often and the results are fairly accurate. In fact, they are often better than the results you would get with a full blown language parser, which has to deal with the nontrivial task of resolving what entity an identifier refers to (often requiring the same about of work as compilation). Yet in our approach, all that is needed is a simple tokenizer, which usually requires writing less than 100 lines of code for each language.

Chava uses Java bytecode to create Acacia databases. From the information stored in the databases (that includes line numbers), we can hyperlink identifiers that refer to the original source code even if the source was not available to Chava at the time of the creation of the database (e.g., source code referred to that is stored in a class library).

3.3. Design Recovery and Visualization Service

In addition to the code analysis, querying, and browsing services, REportal supports a design recovery and visualization service.

Software systems are typically modified in order to extend or alter their functionality, improve their performance, port them to different platforms, and so on. For developers, it is crucial to understand the high-level design of a system before attempting to modify

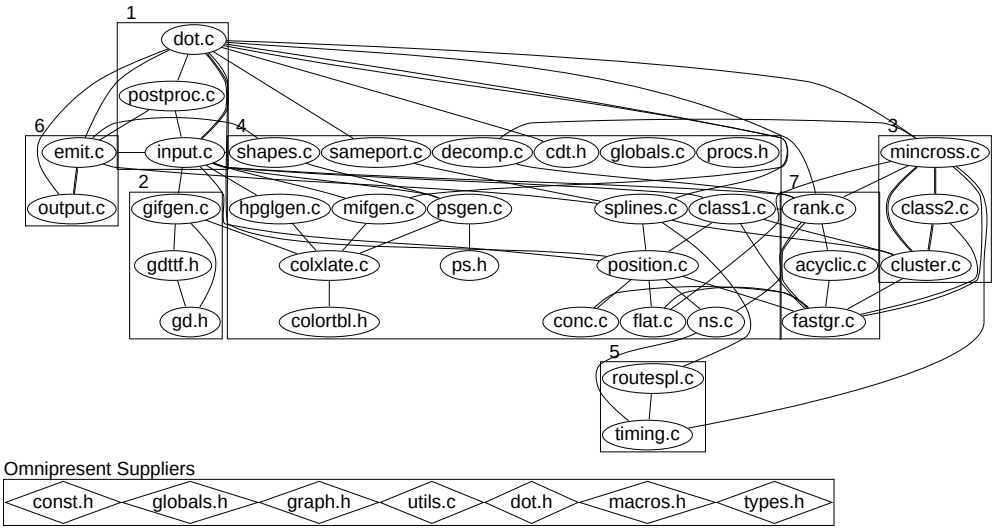

Figure 5. The high-level (abstract) architecture of the dotty system

it. However, the high-level design may not be apparent to new developers, because the design documentation is non-existent or, worse, inconsistent with the implementation. This problem could be alleviated, to some extent, if developers were able to produce design descriptions from the low-level source code.

We have thus developed a tool called Bunch [24, 14, 23] to reverse engineer design descriptions from source code information. The tool partitions the source-level structures into high-level subsystems automatically. Subsystems provide developers with high-level structural information that helps them navigate through the numerous software components, their interfaces, and their interconnections.

The first step in the design recovery process is to extract the module-level dependencies from the source code and store the resultant information in a database. REportal uses the Acacia[10] or Chava [22] tools for this step. After all of the module-level dependencies have been stored in a database, REportal executes a script to query the database, filter the query results, and produce a textual representation of the module dependency graph (MDG). The Bunch tool is then invoked to apply clustering algorithms to the MDG and emit a text-based description of the high-level structure of the systems organization. Finally, REportal uses the Grappa visualization tool [5] to read the output file and show a visualization of the results on the web browser.

In Figure 5 we show the results of applying our tools to the source code of dotty [31]. The model we employed to create the MDG of dotty uses files as modules, and defines a directed edge between two files if the code in one file refers to a type, variable, function or macro defined in the other file. Subsystems are shown as rectangles containing the MDG nodes and edges.

4. REportal Architecture

The architecture of REportal is three-tiered (see Figure 6). The first tier is a web-based user interface, the second tier contains the core servlet, and the third tier contains the services wrapped by the REportal servlet.

To the user, the servlet receives source code and formats the results so that they may be displayed on a web browser. To the lower-level services, the servlet provides authentication, file access, database, and integration services.

In the previous section, we described the third tier portal services. In what follows, we focus on the second tier (REportal servlet) and the security infrastructure.

4.1. REportal Servlet

The services provided by the servlet can be divided into: (a) user interface services to the client and (b) integration services for the encapsulated database, file system, and tools (Figure 7).

File system services can be partitioned into three areas: configuration information, user resources, and temporary space. Configuration information includes location and access information for the database, many parameters governing system configuration and setup. The configuration parameters include: locations of the

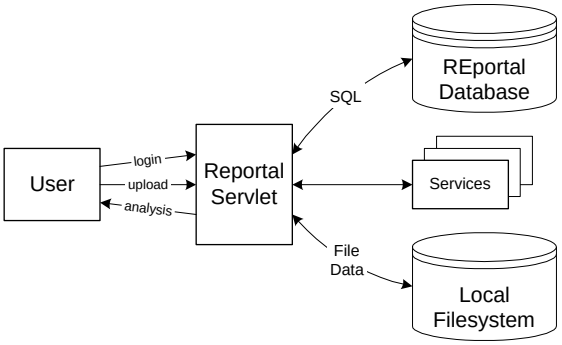

Figure 6. High-Level REportal Architecture

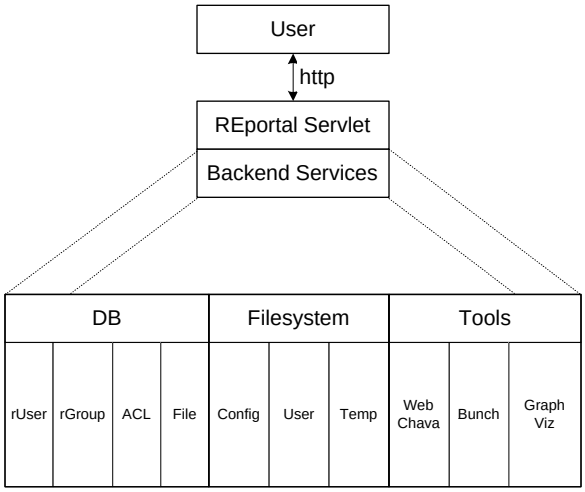

Figure 7. Three-Tiered REportal Architecture

various tools and system utilities, external environment properties, MIME types, *et cetera*. User resources include source files, folders, and analysis meta-data.

Database services provide a coherent interface into the security and authentication services. The database contains four tables, *rUser*, *rGroup*, *ACL*, and *File* and two intersection entities which connect the *rUser* and *rGroup* tables and the *ACL* and *File* tables. The servlet integrates the data stored in the tables into information about the users of the system and the levels of access granted to them.

The tools currently available through REportal are: Bunch, Acacia, Chava, WebChava, GraphViz, and Grappa.

Wrapping of the tools is a multi-part process. First, a permissions check is initiated to evaluate if the user has the appropriate privileges to execute the required external applications. Next, a permissions check is

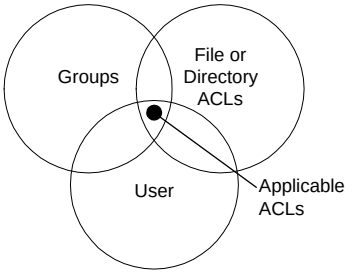

Figure 8. Applicable ACLs to a File or Directory in the REportal File System

made upon the affected files and directories (these checks are detailed in section 4.2). Then, the external tool is executed and the analysis is performed. If its results are directly visible to the user, they are sent to the user as soon as the tool has finished. However, if the results must be displayed using another tool (for example, the results of Bunch must be presented in a Grappa graph), then the results are filtered through that tool before being returned to the user. In the last stage, any temporary files created during the analysis are removed from the file system.

Now that we have introduced the REportal architecture, we continue our discussion with the system's security infrastructure.

4.2. Security Infrastructure

The security infrastructure of the REportal system follows a role-based access control model [33]. In this model, users are assigned to groups. Access permission to files and tools is granted to groups. Thus, through their group membership, users are granted access to the system. The access control model is similar to the permissions model employed in Microsoft's Windows NT [25]. The set of permissions that can be granted to roles are described in Table 1.

Permission to access a file or directory (as described in Table 1) is granted to groups in a set of ACLs. Users in the system are assigned one or more groups based upon their roles. The intersection between these three groups is the set of permissions for the user on the file or directory (see Figure 8). The only exception is the `prohibitAccess` permission which explicitly excludes a group from access to the file or directory. `prohibitAccess` overrides all other permissions and always excludes the user from access to the file.

Next, we describe how REportal can be extended to provide services to other systems.

Permission	Description
`read`	Read from the file
`write`	Write to the file
`execute`	Execute the file
`delete`	Remove the file
`seeFiles`	View the files in the directory
`seeFolders`	View the directories contained within the directory
`createFiles`	Add new files within the directory
`createFolders`	Create new directories within the directory
`modifyFiles`	Modify files in the directory (overridden by `write`)
`modifyFolders`	Modify directories in the directory (overridden by the directory's ACL list)
`deleteFiles`	Remove files from within the directory (overridden by the file's ACL list)
`deleteFolders`	Remove directories from within the directory (overridden by the directory's ACL list)
`setPermissions`	Modify the permissions of files and directories within the directory (effectively granting the group supervisory access to the directory)
`prohibitAccess`	Explicitly deny access to the file or directory

Table 1. REportal Role-Based Access Control Permissions Set

4.3. Extensibility

REportal was designed to be extensible. At this point in the evolution of the system, extending the system is a two part process. First, the new tool is added to the set of tools on the server. Then, the servlet is extended to include the new service in the set of services available to REportal users.

In the future, this process will be automated so that administrators can use a wizard to add new services to the system (for security reasons, the process would most likely require physical access to the system). A more extensive look at the future direction of the REportal system is described in the next section.

5. Future Work

This section outlines our future plans to extend REportal's services, improve its security, and modify its architecture.

5.1. Dynamic Analysis Service

We intend to expand our portal services by integrating more reverse engineering tools into REportal. Specifically, we have developed a framework, called Form [35], to profile and analyze the execution of Java programs (the C/C++ profilers are being developed currently). Form can be customized to create numerous views to support dynamic program understanding. To date, three views have been developed: (a) a view to determine a program's "hot spots", (b) a view to display UML [16] sequence diagrams that show execution traces of object-oriented programs, (c) a view to display UML-like collaboration diagrams that show which objects are created and what messages are sent at runtime.

By integrating Form into REportal we would be able to offer dynamic analysis services in addition to the static analysis services that are currently supported.

5.2. Modifications to the Security Infrastructure

The current version of REportal supports user authentication as a first level of security. Once a user logs into the portal, he only has limited access to a virtual file system sandbox. By limited access we mean that the user can only create project folders, upload files to project folders, delete files from project folders, or delete project folders. In order to provide a dynamic analysis service, REportal must allow users to execute their uploaded code on the site. Thus, REportal will need to be extended to provide a secure context where this code can be executed.

5.3. Modifications to the Architecture

The current architecture of REportal is centralized. That is, the tools and data (i.e., programs) are located on the same machine. This architecture may be suitable for research and education usage, where the software being analyzed is unlikely to be of a proprietary nature. We do not expect software development companies to trust us with their code, although some companies may want to evaluate code that is produced by their vendors and sub-contractors.

Professional software developers are more likely to use REportal if they are not obliged to upload and store

their code on the site. One way this can be facilitated is to use a distributed peer-to-peer architecture similar to that used for Napster. An alternative to the peer-to-peer architecture is to develop system administration tools to help others host their own reverse engineering portal sites. This way companies can host a site behind their firewall and not worry about exporting their code to untrusted foreign hosts on the Internet.

6. Conclusions

In this paper we have shown how several reverse engineering tools can be integrated as a web-based portal site. Our approach has several advantages:

- Reverse engineering services are provided to the user through a consistent web-based user interface.
- Users do not need to be concerned about tool and platform integration or interoperability issues.
- Through REportal, users will always have access to the latest versions of the underlying tools.
- Users are not required to install any software on their computers in order to use the portal services.

The REportal site is available at:

`http://serg.mcs.drexel.edu/reportal/Reportal`

Acknowledgments

This research is sponsored by two grants from the National Science Foundation (NSF): a CAREER Award under grant CCR-9733569 and an Instrumentation Award under grant CISE-9986105. Additional support was provided by grants from the research laboratories of AT&T. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, the U.S. government, or AT&T.

References

[1] N. Anquetil and T. Lethbridge. Extracting concepts from file names; a new file clustering criterion. In *Proc. 20th Intl. Conf. Software Engineering*, May 1998.

[2] The Netcomputing AnyJ Java IDE. `http://www.netcomputing.de/html/main.html`.

[3] Reverse and Reengineering Roadmap `http://www.program-transformation.org/re/`.

[4] AT&T Labs - Research Tools `http://www.research.att.com/sw/tools`.

[5] N. S. Barghouti, J. Mocenigo, and W. Lee. Grappa: A Graph Package in Java. In *Fifth International Symposium on Graph Drawing*, pages 336–343. Springer-Verlag, Sept. 1997.

[6] L. A. Belady and C. J. Evangelist. System Partitioning and its Measure. *Journal of Systems and Software*, 2:23–29, 1981.

[7] S. Bridgeman, A. Garg, and R. Tamassia. A graph drawing and translation service on the www. In *Lecture Notes Comput. Sci. Springer-Verlag*, 1997.

[8] Graph Drawing Server at Brown University `http://loki.cs.brown.edu:8081/graphserver/gds/gds-home.shtml`.

[9] The Western Wares CC-Rider Source Code Analyzer and Browser. `http://www.westernwares.com/`.

[10] Y. Chen, E. R. Gansner, and E. Koutsofios. A C++ data model supporting reachability analysis and dead code detection. In *Proceedings of the European Conference on Software Engineering/Foundations of Software Engineering*, 1997.

[11] Y.-F. Chen, G. S. Fowler, E. Koutsofios, and R. S. Wallach. Ciao: A Graphical Navigator for Software and Document Repositories. In *International Conference on Software Maintenance*, pages 66–75, 1995.

[12] P. Devanbu. Genoa—a language and front-end independent source code analyzer generator. In *Proceedings of the Fourteenth ICSE*, pages 307–317, 1992.

[13] P. Devanbu, D. Rosenblum, and A. Wolf. Generating Testing and Analysis Tools with Aria. *ACM Trans. Software Engineering and Methodology*, 5(1):42–62, 1996.

[14] D. Doval, S. Mancoridis, and B. Mitchell. Automatic clustering of software systems using a genetic algorithm. In *Proceedings of Software Technology and Engineering Practice*, 1999.

[15] P. Finnigan, R. C. Holt, I. Kalas, S. Kerr, K. Kontogiannis, H. Muller, J. Mylopoulos, S. Perelgut, M. Stanley, and K. Wong. The Software Bookshelf. *IBM Systems Journal*, 36(4):564–593, 1997.

[16] M. Fowler and K. Scott. *UML Distilled: a brief guide to the standard object modeling language.* Addison-Wesley, 2nd edition, 2000.

[17] The Freshmeat Web Page. `http://freshmeat.net`.

[18] E. Gansner, E. Koutsofios, S. North, and K. Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214–230, Mar. 1993.

[19] The Rigi Web Page. `http://www.rigi.csc.uvic.ca/Pages/projects/mozilla.html`.

[20] Software Guinea Pigs `http://plg.uwaterloo.ca/~holt/guinea_pig/`.

[21] D. Hutchens and R. Basili. System Structure Analysis: Clustering with Data Bindings. *IEEE Transactions on Software Engineering*, 11:749–757, Aug. 1995.

[22] J. Korn, Y. Chen, and E. Koutsofios. Chava: Reverse Engineering and Tracking of Java Applets. In *Proceedings of the 6th Working Conference on Reverse Engineering*, pages 314–325, 1999.

[23] S. Mancoridis, B. Mitchell, Y. Chen, and E. Gansner. Bunch: A clustering tool for the recovery and maintenance of software system structures. In *Proceedings of International Conference of Software Maintenance*, Aug. 1999.

[24] S. Mancoridis, B. Mitchell, C. Rorres, Y. Chen, and E. Gansner. Using automatic clustering to produce high-level system organizations of source code. In *Proceedings of the 6th Intl. Workshop on Program Comprehension*, 1998.

[25] Microsoft Corporation. *Microsoft Windows 2000 Security Technical Reference*. Microsoft Press, Aug. 2000.

[26] Microsoft's .Net `http://www.microsoft.com/net/`.

[27] H. A. Müller and K. Klashinsky. Rigi: A System for Programming-in-the-Large. In *Proceedings of the 10th IEEE International Conference on Software Engineering*, pages 11–15, Singapore, 1988.

[28] G. Murphy, D. Notkin, and K. Sullivan. Software Reflexion Models: Bridging the Gap between Source and High-Level Models. In *Proceedings of the ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE'95)*, pages 18–28, Washington, D.C., October 1995.

[29] J. Q. Ning, A. Engberts, and W. Kozaczynski. Automated Support for Legacy Code Understanding. *Communications of the ACM*, 37(5):50–57, 1994.

[30] S. North and E. Koutsofios. Applications of graph visualization. In *Proc. Graphics Interface*, 1994.

[31] S. C. North and E. Koutsofios. Applications of graph visualization. In *Proceedings of Graphics Interface '94*, pages 235–245, Banff, Alberta, Canada, May 1994. Canadian Information Processing Society.

[32] D. Rosenblum and A. Wolf. Representing semantically analyzed c++ code with reprise. In *USENIX C++ Conference Proceedings*, pages 119–134, 1991.

[33] R. S. Sandu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, Feb. 1996.

[34] R. Schwanke. An intelligent tool for re-engineering software modularity. In *Proc. 13th Intl. Conf. Software Engineering*, May 1991.

[35] T. S. Souder, S. Mancoridis, and M. Salah. Form: A framework for creating views of program executions. In *Proceedings of the 2001 International Conference on Software Maintainence*, 2001. `http://serg.mcs.drexel.edu/form`.

[36] The Source Forge Web Page. `http://sourceforge.net`.

[37] The Red Hat Source-Navigator. `http://sources.redhat.com/sourcenav/`.

[38] Sun Microsystems' Web Services `http://www.javasoft.com/features/2001/04/track1.html`.

[39] V. Tzerpos and R. C. Holt. ACDC: An algorithm for comprehension driven clustering. In *Proceedings of the Working Conference in Reverse Engineering (WCRE'00)*, 2000.

[40] WebDot Graph Drawing Server at AT&T `http://www.research.att.com/~north/cgi-bin/webdot.cgi`.

[41] T. Wiggerts. Using clustering algorithms in legacy systems remodularization. In *Proc. Working Conference on Reverse Engineering*, 1997.