A Collaborative Demonstration of Reverse Engineering Tools

Margaret-Anne D. Storey
Dept. of Aeronautics/Astronautics
Massachusetts Institute of Technology
77 Mass Ave, 33-407c,
Cambridge MA 02139, USA
+1 (617) 253 4519
mstorey@mit.edu

Susan Elliott Sim
Dept. of Computer Science
University of Toronto
10 Kings College Rd
Toronto, ON, Canada M5S 3G4
+1 (416) 351 9667
simsuz@cs.utoronto.ca

Kenny Wong
Dept. of Computing Science
University of Alberta
221 Athabasca Hall
Edmonton, AB, Canada T6G 2E8
+1 (780) 492 5202
kenw@cs.ualberta.ca

ABSTRACT
This paper describes a collaborative structured demonstration of reverse engineering tools that was presented at a working session at WCRE 2001 in Stuttgart, Germany. A structured demonstration is a hybrid tool evaluation technique that combines elements from experiments, case studies, technology demonstrations, and benchmarking. The essence of the technique is to facilitate learning about software engineering tools using a common set of tasks. The collaborative experience discussed at WCRE involved several peer and complementary technologies that were applied in concert to solve a real life reverse engineering problem. For the most part, the tool developers themselves applied their own tools to this problem. Preliminary results have shown to the research community that we still have much to learn about our tools and how they can be applied as part of a reverse engineering and reengineering process. Consequently, the participants agreed to continue participation in this demonstration beyond the WCRE event.

Keywords
Empirical study, research methodology, tool evaluation, benchmark, reverse engineering, reengineering, program comprehension.

1. INTRODUCTION
This paper describes a working session at the Eighth Working Conference on Reverse Engineering (WCRE 2001), held in Stuttgart, Germany, October 2-5, 2001 on a collaborative demonstration of reverse engineering tools.

A structured demonstration [2] is a hybrid tool evaluation technique that combines elements from experiments, case studies, technology demonstrations, and benchmarking. The essence of the technique is to use a common set of tasks to facilitate learning about software engineering tools. A collaborative structured demonstration, or more briefly a collaborative demonstration, is a structured demonstration where teams are expected to leverage each other’s tools and share results.
architecture that would be better suited for meeting future change requests.

The teams were asked to write reports on their experiences re-architecting the subject software system and to describe how they collaborated with the other teams over a period of six months. The project web site [4] has links to the submitted results.

The rest of this paper is organized as follows. Section 2 elaborates on the motivation and goals for the collaborative demonstration. Section 3 describes the SORTIE case study in more detail and the format of the demonstration, and Section 4 reviews the results from the participants. Section 5 provides some discussion on the lessons learned, and Section 6 summarizes the current status of the collaborative demonstration as it is being continued beyond WCRE.

2. MOTIVATION
The main goals for conducting this collaborative demonstration are:

- to improve reverse engineering and reengineering tools, and develop better ones by comparing and evaluating existing tools;
- to promote collaboration and community-building for reverse engineering researchers and their tools through the use of GXL [1]; and
- to improve tool evaluation techniques.

In the remainder of this section, we elaborate on these goals.

2.1 To Evaluate Related Tools
Although reverse engineering and reengineering tools are designed to solve related tasks in a software reengineering project, they are usually researched, developed, and evaluated in isolation from one another. There has been little work in examining how the tools work together and how they should interoperate from the perspective of an end user. This collaborative demonstration seeks to remedy this shortcoming. Tool designers and potential end users will be able to observe how the best tools and technologies available can be combined to solve reengineering tasks.

2.2 To Promote Collaboration
This tool demonstration promotes collaboration in two ways:

- by providing a focal point for collaboration among researchers and developers; and
- by promoting the use of GXL, the emerging standard format for exchanging data between reverse engineering tools.

Due to the nature of this evaluation, tool designers will need to share data and results with one another, which in turn will encourage them to critically examine interoperability with other tools. We anticipate that participating tool designers will move towards a common understanding on how complementary tools can be used to solve a reengineering task. We also expect that this demonstration will lead to suggestions for advancing interoperability, such as component-based design, mechanisms for coordinating tools, and improvements to GXL.

2.3 To Improve Tool Evaluation Techniques
Carrying out meaningful tool evaluations is a challenge—one that is compounded by working with collaborators with different backgrounds, disparate computing environments, and varying amounts of resources available for their research projects. Consequently, this collaborative demonstration will help us to improve tool evaluation techniques in two ways. First, this collaborative demonstration is part of a series of structured demonstrations where each one has allowed us to explore different combinations of design decisions with respect to timing, location, and the balance of competition versus collaboration. These evaluations will reveal the kinds of results and lessons that can be achieved through the various designs. Second, we also expect to learn about working with a distributed team of co-evaluators. In particular, we are interested in the effectiveness of various supporting technologies and organizational strategies.

3. FORMAT OF THE COLLABORATIVE DEMONSTRATION
In this section, we describe the key elements of the collaborative demonstration. Briefly, teams of researchers and developers attempted to re-architect the SORTIE system using either their own or other tools. At the same time, Storey supervised a parallel attempt to re-architect SORTIE without specialized reengineering tools. A web site and discussion forum were used to disseminate information and to encourage interaction among the participants. Their work culminated in written reports and presentations during a working session at WCRE. Each of these components will now be discussed in detail.

3.1 Subject System and Tasks
The subject system used for this demonstration was SORTIE, a software system developed by the Ministry of Forests in British Columbia and the Institute of Ecosystem Studies in New York State. The SORTIE tool was developed to simulate and visualize forest succession. This tool has evolved over a number of years as the underlying model was developed, refined, and extended.

Mostly, one programmer developed the SORTIE code. It was originally written in C, but has been ported to C++. It runs only on a PC and relies heavily on the Borland C++ compiler and libraries. The current architecture of SORTIE does not easily support changes, such as adding new
variables to the models or adding new visualization features. Any changes that are made invariably have unintended side effects.

The main reverse engineering task for the collaborative demonstration was to propose a new architecture for the SORTIE system to facilitate expected future development. In proposing the new architecture, participants needed to perform a number of subtasks, such as recovering the existing architecture. The following requirements for reengineering SORTIE were posted on the web site during the course of the collaborative demonstration:

> There are two potential uses of the SORTIE program: it can be used as a tool for research exploration in understanding forest dynamics; and it can also be used as a test environment for forest management decisions. In order to support research exploration, the forestry researchers would like to incorporate new relationships and formulas without the assistance of professional programmers. Currently this is not possible, and therefore the code needs to be reengineered to more easily allow these changes. The reengineered program should be based on an object-oriented, extensible, and well documented framework.

### 3.2 Participating Groups

Using the SEWORLD Software Engineering Mailing List and flyers at ICSE (International Conference on Software Engineering) and IWPC (International Workshop on Program Comprehension), we called for tool development teams to participate in the collaboration. We encouraged commercial tool builders to participate and garnered some interest; however, since their primary goal is to deliver products, they have had limited participation thus far. The following teams have been participating in the project to date:

- KBRE Group (Knowledge-Based Reverse Engineering), University of Ottawa, Canada;
- RGAI (Research Group on Artificial Intelligence), Hungarian Academy of Sciences, University of Szeged;
- Rigi Group, University of Victoria, Canada;
- SWAG (Software Architecture Group), University of Waterloo, Canada;
- SCG (Software Composition Group), University of Berne, Switzerland; and
- Visualization Research Group, University of Durham, UK.

### 3.3 Parallel Manual Project

A graduate student at the University of Victoria, Sachen Gendron, has been also been working to re-architect SORTIE primarily using manual techniques. This parallel manual project serves two purposes. First, it serves as a basis for comparison for the results from the other tools. The basis for comparison allows judgements to be made about the performance of other tools. Second, this experience in re-architecting SORTIE allowed Gendron to act as a domain expert for this evaluation. She is working closely with the forestry researchers in B.C. and New York. Participants were encouraged to contact her if they had further questions about the code base or user requirements.

### 3.4 Web Site and Forum

Since we were trying to foster collaboration among a variety of participants with different computing environments located across the world, we used the Web and email as the primary mechanisms for interaction. All the organizational information was available on a web site and the SORTIE source code was downloadable with a password. We also used a mailing list and forum where participants could send each other email and share files. We strongly encouraged participants to post results, questions for each other and the domain expert, lessons learned, problems, and progress on this forum.

### 3.5 Results and Reports

All participants were asked to submit a written report, detailing the results of their analysis in advance of WCRE. We asked the participants to use the following sections for their reports:

- an introduction to provide a brief description of their tool(s) and background on the team members;
- an experience report to describe the steps that were taken to analyze the software system, time spent on each step, and any difficulties and surprises encountered;
- a discussion on collaboration partners, i.e., peer and complementary tools used in the demonstration, including the steps required to make the tools work together; and
- a solution to the reverse engineering tasks, specifically documenting the existing architecture of the SORTIE system and proposals for the new architecture and how this new architecture should be achieved.

In addition to the reports, the teams were to present their results in the working session at WCRE.

### 4. RESULTS

In this section, we give a glimpse of the results described in the reports and the presentations at WCRE. For further details and references, please refer to the reports posted on the web site [4].

- The KBRE Group developed a parsing tool generating GXL and DMM. They also employed approaches to clustering, using agglomerative hierarchical algorithms. They experimented with displaying the results on the
web using third-party graph-drawing tools such as the AT&T GraphViz tool.

- The RGAI team used the CAN/Columbus parser/analyzer for analyzing the source code.
- The Rigi Group experimented with both the Rigi C++ parser and the TkSee C++ parser. They visualized both sets of parsing results using the Rigi graph editor.
- The SWAG team used their CPPX parser for analyzing the software code. CPPX is an Open Source, AST-level fact extractor for C++ that relies on GNU g++ as a front end. They used the PBS (Portable Bookshelf) tool suite for visualizing the results.
- The SCG team applied their toolset, which consisted of elements from SNiFF+ (a commercial tool), Moose, and CodeCrawler.
- The Visualization Research Group used GraphTool for visualizing the GXL code generated by the TKSee C++ parser.

The most striking result is that the different teams used the collaborative demonstration for a variety of purposes many of which we did not anticipate. The KBRE Group used SORTIE as a test case to help debug their new C++ parser and visualization infrastructure. Similarly, RGAI and SWAG also used SORTIE to help test their C++ parsers. SCG used the project as a reverse engineering exercise for training graduate students. The Visualization Research Group used the demonstration as an opportunity to implement GXL support in GraphTool. The Rigi Group also used the project to train graduate students and debug their tools.

In the working session, the participants were asked to make a series of joint presentations to present their own results and reflect on the results of the other teams. We asked for joint presentations to encourage further collaboration and learning, and to make it easier for others in the research community to understand the significance of the different results. The presentations were ordered as follows:

- **Joint Presentation on Parsing** by Holger Kienle from the Rigi Group, University of Victoria and Rudolf Ferenc, RGAI team, University of Szeged
- **Joint Presentation on Parsing** by Davor Svetinovic, SWAG team, University of Waterloo and Tim Lethbridge, KBRE Group, University of Ottawa
- **Joint Presentation on Visualization** by Tim Lethbridge, University of Ottawa and Claire Knight, Visualisation Research Group, University of Durham
- **Parsing and Visualization Experiences** by Michele Lanza, SCG team, University of Berne
- **Domain Expert Presentation**, Sachen Gendron, University of Victoria.

These presentations are also available for viewing on the web site [4].

The presentations were followed by a group discussion where the general consensus was that many of our tools are not yet at the stage where they could be transferred to industry, and that some of them could only be effectively used by expert reverse engineers. There was also the issue that more integration of static and dynamic analyses was needed. The issue of general versus specific tools was raised, in that many of the tools in the exercise were not designed for this particular kind of system (namely Borland C++ and its proprietary libraries).

In general all the participants agreed that much was learned through this exercise and that we should continue with the demonstration over the following months. Gendron found the report by the SCG team to be the most useful and that their tools could have helped her during the manual effort. She also mentioned that the forestry group in New York had recently employed a new programmer and that a new version of the source code could be made available. This was of interest to researchers developing tools and techniques to assist in evolution.

5. LESSONS LEARNED

To discuss the lessons learned thus far in the collaborative demonstration, we return to our three main goals:

- to compare related reverse engineering and reengineering tools;
- to promote collaboration among researchers and their tools; and
- to improve tool evaluation techniques.

Since the collaborative demonstration is ongoing, we give a preliminary discussion of the lessons learned to date.

5.1 Comparing Related Tools

As anticipated, the collaborative strategy allowed comparisons to be made among tools in the same phase (peer technologies) and among tools in different phases (complementary technologies). Of the teams that presented at WCRE 2001, we had strong representation by tools from the parse and document/visualize phases and this distribution was reflected in the groupings for the joint presentations.

- **Parsers**: CAN/Columbus (RGAI), CPPX (SWAG), Rigi, and TkSee/SN (KBRE).
- **Document/Visualize**: CodeCrawler (SCG), GraphTool (Visualisation Research Group), PBS (SWAG), and Rigi.

While all teams used analysis tools, these tools did not figure prominently in their reports or presentations. Comparisons of these tools will need to be done indirectly, by examining the results of the reverse engineering or
reengineering. We also had one reengineering tool participate (Moose from SCG), but it was not used to modify the source code.

All of the parsers handled C++, but they differed in their level of detail and output format. CAN/Columbus and CPPX both produce facts at the AST level in GXL as well as their local formats. Rigi emits RSF (Rigi Standard Format) and TkSee/SN emits GXL, but they work at the middle level (sometimes called the external declaration level). TkSee/SN was the first parser to produce a set of facts for SORTIE in GXL; consequently, a large number of downstream tools used data from this parser.

All the tools from the next phase were visualization tools. The appendix contains several snapshots from the visualizations in the reports as well as the views manually created by Gendron. It is interesting to note the different kind of views automatically generated by the tools to those created manually. The manual views capture the domain dependent concepts and more of the dynamic behavior of the software, whereas the automatic views capture more about the source code constructs, relationships and statistics. This is not surprising, as the manual views were created using information from the domain experts, as well as detailed examinations of the source code and the run time behavior of the program.

5.2 Promoting Collaboration

Although there was not as much collaboration as we had hoped, we were still successful in fostering interaction among the researchers and the use of GXL between their tools. Participants interacted during the demonstration in a number of ways: seeking out the required collaboration partners, sharing their results, and preparing joint presentations for the working session. All of the data exchanged between teams employed GXL. The Visualization Research Group used this demonstration as an opportunity to add GXL handling capabilities to GraphTool. The Rigi Group implemented a GXL to RSF converter. In addition, the SCG team provided metrics to be used and validated by other teams.

The main impediments to collaboration were technology and timing. Initially, we used a free, public service for hosting discussion groups, eGroups. Over the course of this project, the company was acquired by Yahoo, which resulted in the software becoming much more cumbersome to use. We hope to remedy this problem by setting up a Wiki [5] for the next phase of the project. Also, the various teams worked on SORTIE at different times, so people often did not have answers when others needed them. For example, the teams developing CPPX and CAN/Columbus were still implementing their parsers and did not produce data until late in the process. Similarly, our domain expert was not widely available until the later stages of the project and consequently few participants contacted her.

Finally, we do not yet have enough experience to comment on an infrastructure for community-wide sharing of tools. We hope to gain insights at this level in the next stage of the collaborative demonstration.

5.3 Improving Evaluation Techniques

As mentioned earlier, the collaborative demonstration is part of a sequence of usability studies and tool evaluations conducted by the authors, the most recent of which are the structured demonstrations.

The collaborative demonstration builds on this previous work by using a common subject system and tasks to evaluate tools. Unlike our previous evaluations, we used a subject system that was currently in use and a task that was set by the owners and users. Consequently, SORTIE presented a series of unusual problems to the participants. SORTIE is tightly integrated with an integrated development environment and graphical user interface builder, and runs on Windows. Both of these characteristics posed problems for teams who normally apply their tools on Unix to Open Source software.

When selecting a subject system, it is difficult to choose one that is accessible to both research and commercial tools, while at the same time is novel enough to point out new research directions. This dynamic is most obvious when deciding on the desired size of subject system: it should be sufficiently large to showcase the more mature reengineering techniques, but not too large to test techniques still under development that may not yet scale. SORTIE was relatively small in terms of size, but it was able to illustrate strengths and weaknesses in the tools used.

The collaborative demonstration differs from a previous structured demonstration because it was a non-collocated, and asynchronous (as opposed to the collocated, synchronous structured demonstrations that were held previously[2]). While these changes affected the intensity of the interactions, we believe that we are still able to obtain the most significant benefits of the structured demonstration format: evaluation, education, and community building. At this stage it is not clear what are the key characteristics responsible for this success, but we plan to pursue this work for as long was we, as a community, continue to learn from it.

6. CURRENT WORK

The working session at WCRE was expected to be the conclusion of this structured demonstration, but there was a general desire by the teams to continue their work on SORTIE. Many of the teams had difficulties parsing or had to spend extra time modifying their tools to support GXL. Consequently they did not have time to fully pursue the main task of re-architecting SORTIE. Furthermore, there was a desire to continue the collaboration in order to learn more about each other’s tools.
As mentioned in Section 4, the SORTIE developers have a new version of the source code available as a new programmer made some changes and improvements to the code base. Their next step is to start the reengineering process. They will work closely with Gendron during the reengineering phase. She, along with the group in New York, are excited about this effort and hope that the input from this demonstration will be a significant aid as they re-architect their system.

This next phase of the collaborative demonstration will conclude with a similar working session to this one. The exact venue and date will be posted on the web site [4]. We are very interested to have new tools and research teams participate in this phase. In particular, we are interested in commercial tools, which may be applied by the developers or by interested researchers wishing to learn more about these tools.

ACKNOWLEDGMENTS
We are very grateful to all of the participants for taking much of their valuable time and contributing to this event: Holger Kienle, Xiaomin Wu, Timothy Lethbridge, Sergei Marchenko, Claire Knight, Michele Lanza, Gabriela Arevalo, Daniel Schweizer, Daniele Talerico Andrew Malton, Davor Svetinovic, Thomas Dean, Rudolf Ferenc, Árpád Beszédes, Tibor Gyimóthy, and especially Rainer Koschke for having faith in this effort and scheduling time at WCRE 2001 for our presentations.

REFERENCES

Appendix
This appendix contains snapshots generated by the research teams employing visualization tools. The reports on the web site [4] describe the purpose of the views and how they were generated.

Figure 3: This view is from the KBRE report. It shows a snapshot generated by AT&T GraphVis tool, showing the Clustering of classes by Types Referred.

Figure 4: This picture was generated by the GraphTool from the Visualisation Research Group, University of Durham.
Figure 5: This picture shows snapshots from the Rigi report. The top view is the initial graph that was generated by the Rigi C++ parser, which had 8518 nodes and 23060 arcs initially. The bottom view was obtained after several operations were performed by the reverse engineer in Rigi.

Figure 6: These views were generated by the Code Crawler Tool, by the SCG Group, University of Berne.
Figure 7: These views were generated by Sachen Gendron by following a manual reverse engineering process.

1 M. Storey is currently on leave from the Dept of Computer Science, University of Victoria. Email: mstorey@uvic.ca.