UCRL-ID-153221

Research Report on Feasibility Study of Building a QT Gui Testing Tool-AX Program Code Group Computer Science R & D Project (U)

Benjamin T. Grover

May 5, 2003

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

DISCLAIMER

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

Research Report on: Feasibility study of building a QT GUI testing tool, *AX-Program Code Group Computer Science R&D Project*
Benjamin T. Grover, PMesh Project

Participating Employee:
Benjamin T. Grover

Start Date: November 15, 2002 **End Date:** March 1, 2003

Background:

See Appendix A, Proposal

This project addressed the need for a GUI quality assurance testing tool for the PMESH project. We currently do not have any automated method for testing our GUI. Commercial GUI test tools do exist, but only one exists to test Qt. This project consisted of building an in house Qt GUI test tool and trying out the one commercial tool that tests Qt.

Proposal Summary:

The main goal of this project was to determine if a tool could be built to test Qt. In determining the feasibility of building a tool the following requirements needed to be researched:

1. Determine if the underlying Qt signal/slot architecture could be leveraged.
2. Research how much impact implementing such a tool would have on existing code, i.e. how much extra code would need to be inserted to use the tool.
3. Determine with the above information if a tool could be built.

With the above steps completed, the information needed to make a decision on building a tool could be made. Armed with this information I felt I could make a more educated decision on the possibility of building a tool.

Project Overview:

This project was divided into two main steps. The first step was to understand the underlying Qt source code much better. The second step was to build a small prototype that I could use to test ideas. The first step was actually much shorter than I had originally anticipated. Understanding the underlying architecture of Qt only took about two weeks. After studying the architecture of qt and working with the support people at Trolltech, the company that develops Qt, I found a way to test Qt.

In my proposal I figured I could use the signal/slot architecture of Qt to test the GUI, but I found that I could only directly use the slots. The signals in Qt are protected methods of the class that they reside in. This meant that if I could manage to find out what signals were being used for a given event, such as a button press, I still could not re-emit that

signal. After hitting this roadblock I found a way to monitor all events that were happening in Qt via an event filter. This filter enabled me to play big brother and watch everything that was going on in the GUI. Events are stand alone objects and can be emitted without any ownership violations. In fact re-emitting events was a better idea than using signals, because an event is really the point of entry from the underlying window manager to Qt. When Qt receives an event from the underlying system it emits a signal. With this capability and information I was ready to build a prototype.

The prototype I built consisted of a small application that would print items out to standard output upon the press of a button. The prototype also had an object window that would allow me to see the events as they happened as well as give me the ability to replay events and save the events to a file. This prototype allowed me to take an even deeper look into what was going on in Qt. The prototype did not have all the functionality needed for a full scale tool such as full automation, but it did give enough information to realize that building a tool would be feasible. My days of research usually consisted of trying to get the test application to do things in many different ways, and then see if I could re-create the same events I had just executed. In some cases it could, in other cases I found out things I had overlooked or did not understand. I would then remedy those cases and move on and try something new.

In the middle of testing my prototype (January 2003) I heard of a newly released tool called KDRunner. KDRunner is made by Klarälvdalens Datakonsult in Sweden. KDRunner basically did the same thing that I had set out to do: test Qt effectively. Since this was a feasibility study, part of the study was to find other tools that may work, instead of building my own. I downloaded and tested an evaluation version of the tool. The tool worked relatively well, but was no better than the tool I had already developed. In fact it wasn't able to recreate some events that my tool could. I decided to stick with the tool I was working on. KDRunner may improve as it gets more users and matures a little, but I felt more comfortable using the tool I had developed, and I felt that my tool could test better then KDRunner.

Since I was able to make a lot more progress on this tool than I had originally anticipated, as a last test I hooked up the prototype to the mesh generation tool Draco. I wanted to see if I could actually use this tool for what it was intended for. After working out a few glitches it was able to replay multiple scripts to Draco.

Project Summary:

This project was very successful. I accomplished everything I intended to do. I learned and understood the inner workings of the Qt library enough that I could build a simple tool that could leverage some of the information in Qt to test the GUI. I was also able to find a tool that was commercially available to test Qt GUI's. These two things were the main goals of this project. Therefore I consider it a success. In fact I was able to progress farther with my prototype testing then I had originally planned.

Future Work:

There is still much to be done to have an effective testing tool for Qt GUI's. Below is a list of tasks that still need to be accomplished:

- Build on the prototype testing tool to make it more robust, user friendly and functional.
- The prototype currently records some actions such as mouse coordinates. This is not desirable. The final product should record user intentions that are carried out via mouse movements. Along this same line the developer should be able to enhance the tester for special widgets so it can record the correct behavior.
- Build into the code the ability for the developer to carry out pre and post conditions tests as they test the GUI. {Partly Done}
- Look into making the output script a little more readable. Currently, the script is tab delimited text. This can be hard to read.

References:

www.trolltech.com
www.klaralvdalens-datakonsult.se

Appendix A: Proposal

Proposal to look into the feasibility of building a Qt GUI testing tool
Benjamin T. Grover, PMesh Project

Background:

Graphical user interface (GUI) software testing is an inherently complex problem. On the website http://www.csst-technologies.com/genericGraphical_User_Interface_Testing.html some of the problems of GUI testing are outlined. Some problems are:

1. GUI's are an event driven environment, meaning any number of event sequences can happen depending on what the user desires, there is usually no one logical flow of events.
2. GUI's have multiple input media. The mouse and keyboard can be used either independently or together.
3. GUI's are usually built to function on multiple platforms. These allows simpler software development, but each system handles events differently. Consequently each system may open windows in different areas of the screen compared to another system.

While I was in college I worked as a software tester at Novell. I was hired to test the functionality of our GUI's that were in use at Novell. I manually tired to test every facet of the GUI by using a set of written test cases. I had to open dialogues, press buttons, resize windows and do many other things to make sure our GUI worked correctly. The only alternative that I know of to testing a GUI is to get an automated tool that records a tester's keystrokes and mouse movements. These type of programs work OK if nothing new is added to the layout of the GUI, or new functionality is not added. If anything changes on the GUI the mouse movements and keystrokes must be re-recorded. It is an inefficient process at best.

In order to solve the cross platform (linux, aix4, solaris, etc.) development problem that we face here at the lab my group (Pmesh) uses the Qt software library to develop and build our GUI. Qt is simple to use, it is open source, and is written in C++. For these reasons Qt is a popular tool both inside and outside the Lab. Some other groups that us Qt: VisIt project in B division, and it is being considered by the Arachne project in A division.

Qt uses a signal and slot architecture to run the GUI. Basically this means that when an event happens, e.g. Button press, menu item selected, etc., a signal is sent. This signal is received by a slot which executes commands when it receives the correct signal. This signal/slot architecture can be utilized to test the GUI.

Proposal:

I would like to look into the feasibility of creating an automated testing tool for Qt. This is not a proposal to create the tool, but instead a proposal to spend time getting all the information and ideas I need to write a proposal asking permission to actually build this testing tool. Some questions I will answer as I gather information and do research:

*1.*Do any good testing tools already exist? *I have already looked around quite a bit on the internet and have found some tools out there but they mostly record keystrokes and mouse movements. I have not been able to find any good testing tools that leverage Qt's signal and slot architecture. I do know there is interest in this though, because I have seen many message threads talking about building a good testing tool for Qt. There are some tools out there that are very similar to what I would like to build for Qt but they use the Motif library, not Qt.*

*2.*How can the testing tool be built, so the developer doesn't need to test each GUI manually? *In this case I have a plan to leverage the signal and slot information inherent in Qt. I hope to eventually make a tool, that captures all the signals for a given dialogue and fires the signals and checks to make sure all the slots execute correctly. The signals will be recorded in a script that can be run. I hope that this program requires very little developer time, i.e. they can down load it and link it with their program and it basically does the rest.*

Importance to the Code Group:

This project if it succeeds, will be very helpful to the code group as well as anybody else that uses the Qt libraries. It will address the need of being able to test all aspects of the GUI in an automatic way. This will insure a higher quality software product as well as free up developer time to do more important tasks.

Involvement:

I will mostly be involved in doing this research. I will also collaborate with some of my colleagues at the lab as well as some at Sandia National Labs.

Time:

This research will take one quarter, probably 2 hours a day.

Milestones:

1. Find the information I need about the underlying Qt framework specifically the signal and slot information.
2. Determine if the signal and slot information can be extracted easily from the code being tested.

3. Determine if a lot of new code will need to be inserted into the existing code to extract the signal/slot information.
4. Determine if the GUI could be tested well with the signal slot information found.
5. Decide on the feasibility of submitting another proposal to build an actual testing tool.

Reference:

www.trolltech.com (Qt web page)

University of California
Lawrence Livermore National Laboratory
Technical Information Department
Livermore, CA 94551