TESTING GRID SOFTWARE: THE DEVELOPMENT OF A DISTRIBUTED SCREEN RECORDER TO ENABLE FRONT END AND USABILITY TESTING

Florian Urmetzer, Gareth J Lewis, Vassil N Alexandrov
Advanced Computing and Emerging Technologies Centre, University of Reading

Abstract

Ongoing research has shown that testing grid software is complex. Automated testing mechanisms seem to be widely used, but are critically discussed on account of their efficiency and correctness in finding errors. Especially when programming distributed collaborative systems, structures get complex and systems get more error-prone. Past projects done by the authors have shown that the most important part of the tests seem to be tests conducted in a test-bed carried out with human testers. However, reconstructing errors was often impossible. The researchers have developed a distributed screen recorder, which enables the tester to record screens in different locations. The playback is synchronous and can therefore be used to easily reconstruct moments in time, before errors have occurred. Additionally, the screen recorder allows conducting usability tests of distributed applications, by recording webcam pictures of the users involved in the test. Therefore the application will make front-end and usability testing of collaborative grid applications easier and more efficient.

INTRODUCTION

In academic as well as industrial collaborations more and more distributed software are used to support collaborations between different entities in projects and companies. This trend is based on the theory that collaboration gives competitive advantages to its users [1]. Further, the trend is interconnected to the growth of use of Grid technologies and virtual organisations [11]. In return this, means that there is a growing need of software to support this movement, which is reflected in a growing number of developments of such systems [e.g. 2, 3]. These software products are complex, simply based on the fact that their client computers are distributed. Because of this distribution therefore the development of computing software will get more complex. This complexity was not reflected in testing procedures until the authors recent paper has discussed a basic framework for testing collaborative software [4].

This paper contributes to the discussion of testing grid software in two ways. The first is to discuss how screen recorders can be used to enable backtracking of the users screen interactions. This will therefore deliver an easy way to reconstruct errors invoked through the user interface. The second contribution is to show how usability studies can be conducted in distributed environments. This method is based on the use of screen recorders to track multiple users' interactions working in distributed places on one task including a video and audio transfer of the user.

Out of both needs the authors have developed software to enable the recording of media streams (e.g. screens, cameras and audio) coming from distributed locations called 'Screen recorder and distributed debugger' ScRaDD. These locations can be multiple users working on one task collaboratively over networks.

PROBLEM DESCRIPTION

When developing software there are situations where agents involved in the development process are interested having visual access to the screens of users. This mainly occurs because of two reasons: first, an interest due to error tracking and second because of interest in testing the usability of the software.

The first is important for example when errors occur and it is important to back track the events invoked by the user. When using screen recorders, these errors are then easy reconstructed by looking at the video footage of the screen. Therefore the last user interactions can be recreated and the error easily reproduced. This is commonly used when testing applications and is for example described by Whittaker [6] with the difference that he uses a person to remember the last steps the user has done. This is however virtually impossible when the users of the software are in distributed locations. This includes that when systems are collaborative the error invocation could be due to multiple user involvement in creating the error within the software. The authors found in past developments a screen recorder being able to record and play back multiple screens can be essential for such error reconstruction.

Also it is useful to be able to track the state of each computer involved in the collaborative work task. This means that it can be useful to be able to analyze the performance and to get information for debugging. As performance indication, the memory and the CPU usage where found important trough out the work. In detail the debugging information contains stack details, operating system messages e.g. signals, network usage and a measure of functions used from the programme in runtime were found to be useful. The information allows evaluating which functions are important to be optimized for high programme efficiency.

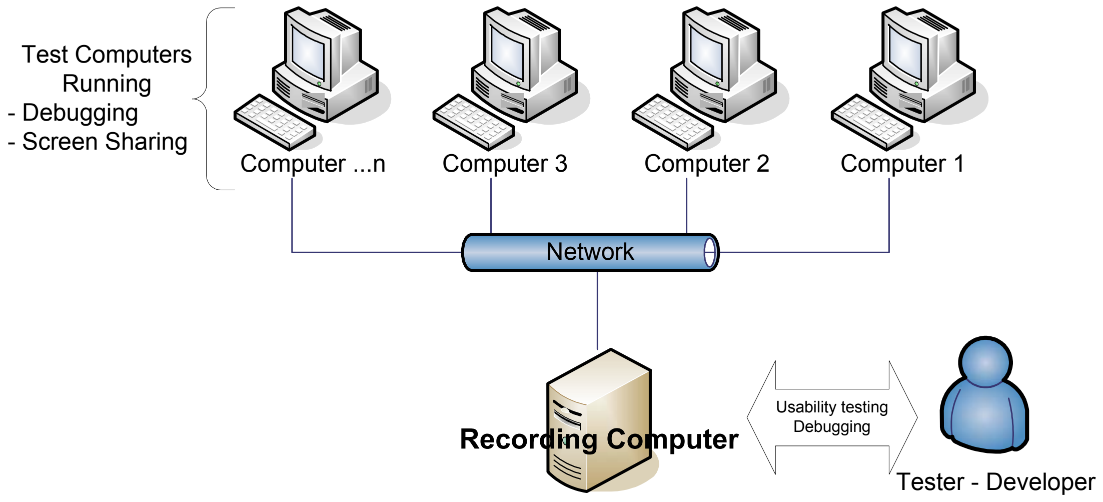

Figure 1: visualization of the networking setup of the distributed screen recorder.

The second example of the use of such recording software is when usability testing distributed applications. Usability test methods include the observation of people while they are using software to be able to conclude how they fulfil tasks. This includes in most cases either to watch people and to take notes or to use a camera and a screen recorder for later evaluation [7-10]. This means that observing people using distributed software needs the ability to record and playback multiple screens and multiple cameras. Critiques argue that taking video footage of usability studies should not be supported, because of time constraints to analyze material, intimidating users and therefore introducing bias towards the study and finally that the effort to set up equipment is not worth the outcome [9]. These critiques apply to single user applications. On the contrary the authors found that when usability tests are conducted in collaborative distributed environments or in virtual environments, studies should be supported with such visual aids. However the time needed to analyze such material should not be underestimated.

THE FRAMEWORK

There are three parts to the software setup. The first is to make screens visually available over the network. In addition there is the option to transfer a camera image via the same network. Secondly there is the option to visualize the state of the computers over the same network. Finally there is the recording of all the images (screens, video and audio) on a server or computer (see fig. 1). These streams are then playable by the tester.

The researchers attempted to work with as many standard tools as possible to fulfil this task. However not in all cases appropriate standard software where available.

The ACET Centre has developed software to allow the sharing of screens to multiple clients via a network. This software was taken and adapted for the task. The recording will be done using custom produced software.

SHARING OF INFORMATION

There are three types of information that are transferred to the recorder. First the screen of the computer, second the machine state information and finally there is the option to transfer one or more attached cameras to record the user behaviour including sound.

The screen is grabbed, and then transferred in video format using the mpeg 4 compression to reduce the information sent and prevent the network from flooding. The choice for mpeg 4 was based on the availability and rather good compression rate. Additionally within the standard there are several options available to adapt and enhance the compression.

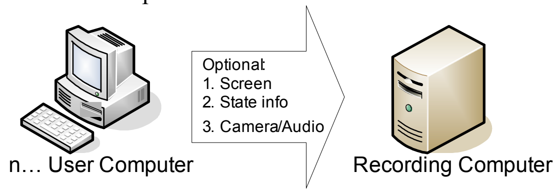

Figure 2: Sharing of information.

The machine state is transferred using textual information. The researchers have chosen to use a text based transfer of this information, because of three advantages. First text based information takes less storage space and therefore as well less networking capacity rather than visually. Secondly textual information is easily searchable, which makes the process to find for example situations where CPU usage went over 90% easier. Finally there is the option within the mpeg4 standard part 17 to transfer text as subtitles, which will make the attachment of the text to video streams easier. With the present version the transfer of the state information is done as an attached to the frames.

NETWORKING

The ScRaDD allows both developers and testers to have a better understanding of how an application performs. The tester is able to inspect the video of the application from all the participants to identify the actions that occur to cause an error or investigate the usability of an application. The developer of the application is able to gain valuable information about the application from debugging and profiling information from each participant. ScRaDD uses MPEG-4 for the video and streams both video of the screen and the debugging/profiling information to the developer from the slave machines using the Real-Time Protocol (RTP). Using RTP means that we can send the data as separate streams. This has the advantage that if the developer only requires the debugging and profiling information and not video of the screen and webcam then this can be specified. The synchronisation of the debugging text and the video ensures that the debugging information is appropriate to the application video and webcam. The profiling information allows developers to investigate the time spent in specific functions and by associating this with the screen recorder it is possible to see how different participants can impact on each other. The tool currently involves creating a session to which each of the slave machines connect. Once connected the slaves stream the data to the machine that is specified as the master, it is possible to have more than one master specified to allow more than one developer/tester to receive the data. The master receives the video and text data which, can be viewed in real time or saved to playback at a later stage. The master user can review the video streams and compare the debugging information of each participant. ScRaDD uses current debuggers and profilers to obtain the debugging information. We are currently looking at GDB and gprof and will extend this to be available for different platforms.

RECORDING

The recording is done using a custom made programme that writes the incoming information to disc. Depending on the size of the project, meaning the number of participants or the number of screens that need to be recorded, a choice needs to be made on the equipment used to store and retrieve the data from the device. During the basic tests the authors have used a desktop pc for the recording; however the testing was based on a proof of concept, and therefore limited to the recording of four desktops. Further tests are planned to be done in the near future.

When the project size grows disc sizes have to be scaled up and it has to be made sure that networks are not flooded with data.

FUTURE WORK

The software exists at the time of the paper as a working prototype with not all, but most functionality. The development is ongoing and the authors are keen to get the software to stable release.

Future work includes:

-To integrate current debuggers and profilers for different platforms - ScRaDD is intended to be used on different platforms, to achieve this it must be able to take advantage of current debuggers/profilers on the different architectures.

-An investigation into the use of ScRaDD between slaves and masters separated by firewalls - ScRaDD has only been used within a local setting for testing and debugging of applications. Scaling this model might involve using salves and masters that are separated by firewalls.

-Using ScRaDD for testing a large number of participants will lead to an increased load on the masters, we are looking at ways to reduce this load, either by providing a portal based master or allowing masters to specify when video should be sent e.g. if an error occurs on a slave then the previous x secs of video and debugging information should be sent to the master or receiving video from only a subset of the participants.

CONCLUSION

The authors have demonstrated as proof of concept that it is possible to record multiple distributed computer screens and information about states of machines and webcams. These recordings can then be used in several ways. This paper takes distributed usability testing and error detection as examples of use.

ACKNOWLEDGEMENTS

The authors of this paper would like to be grateful to their team. The ACET centre has up to now self financed the developments of the screen recorder.

REFERENCES

[1] Dyer, J.H., Collaborative advantage: winning through extended enterprise supplier networks. 2000, Oxford: Oxford University Press. xii, 209.

[2] Lewis, G.J., et al. The Collaborative P-GRADE Grid Portal. in International Conference on Computational Science (3) 2005. 2005. Atlanta, US: Springer.

[3] AccessGrid. Access Grid: a virtual community. [Internet] 2003 13 March 2003 [cited 2006; http://www-fp.mcs.anl.gov/fl/accessgrid/]. Available from: http://www-fp.mcs.anl.gov/fl/accessgrid/.

[4] Urmetzer, F., et al., Testing Collaborative Software: The development of a testing framework for collaborative software, in Cracow Grid Workshop. 2005: Cracow Poland.

[5] Bajwa, H., W. Xiong, and F. Maurer. Evaluating Current Testing Processes of Web-Portal Applications. 2005 [cited 2005; http://ebe.cpsc.ucalgary.ca/ebe/].

[6] Whittaker, J.A., How to break software: a practical guide to testing. 2003, Boston, Mass.: Addison-Wesley. 1 v. +.

[7] Faulkner, X., Usability engineering. 2000, Houndmills, Basingstoke, Hampshire: Palgrave. xii, 304.

[8] Kuniavsky, M., Observing the user experience: a practitioner's guide to user research. Morgan Kaufmann series in interactive technologies. 2003, San Francisco, Calif.; London: Morgan Kaufmann. xvi, 560.

[9] Nielsen, J., Designing web usability: [the practice of simplicity]. 1999, Indianapolis, Ind.: New Riders. xiii,419p.

[10] Nielsen, J., Usability engineering. 1993, Boston; London: Academic Press. xiv, 358 p.

[11] Foster, I., Kesselman, C. and Tuecke, S., The Anatomy of the Grid: Enabling Scalable Virtual Organizations. 2001, International Journal on Supercomputing Applications, 15 (3).