Project Evaluation
3D Heap Visualisation

Aviv Beeri, Briony Goldsack, Ying Jiang,
Oliver Myerscough, Anna Thomas, Eleanor Vincent

November 25, 2013

1 Introduction

An object-oriented software application creates complex structures on the heap during its lifetime. Debugging object-oriented software often involves thinking about how the heap evolves as a program runs. The aim of our group project is to design a tool which supports visualisation of the heap of a running Java program as a 3D scene which can be navigated by the software developer by moving around as if in a first person shooter game.

This report aims to describe how we have quantitatively evaluated the work we have produced and the techniques we have employed to measure and analyse our results. We will discuss the particular qualities of our project that are most important in carrying out our evaluation, the metrics we have used, and the data we have gathered.

2 Project Evaluation

There are several qualities within our project that are particularly important when carrying out evaluation on the Heap Visualisation tool and assessing its success as a product. Firstly, we feel that the performance, intuitiveness and functionality of the debugger aspect is very important to our project as this is the foundation of our tool and should behave smoothly and reliably. We feel that another important feature of our project is the graphical visualisation of the heap itself. It was imperative that the visualisation had the qualities such as being informative to the user, intuitive to use and visually appealing so that we can satisfy our stakeholders as identified through the requirements and customer processes.

We have employed a few different methods to evaluate what we have produced and the assess the important qualities we view as necessary to the project.

3 Testing

Our automated build process and test suite has enabled us to continuously and quantitatively assess the quality and correctness of our code as we have progressed with the project. The Travis Tests indicate the number of failing builds.

4 Data Collection

To further improve on the user experience of our tool, we could use analytic tools to record some quantitative data on how users interact with the product. These would include the amounts of idle

time between mouse clicks, recording the path of a user through the program, where they get stuck and perhaps even using some A/B testing to try different program layouts.[4] This data would allow us to see areas that would need improvements, but the disadvantage is having to actually implement the mechanisms for collecting these analytics, so it is unlikely that we will use this particular technique to gather feedback from potential users. However, after the project is completed, we could use it for further improvements.

4.1 Usability Feedback Options

- Oliver can test with his PPT group (once we have something to show). Check our GUI is intuitive enough for inexperienced programmers.
- We asked our supervisor to try out our GUI (at a very early stage). He gave us some useful feedback on labelling buttons etc.

4.2 Questionnaire

If the project is to be a success, it is vital that all customers we have targeted find the system intuitive and useful. This is challenging as they come from a wide range of backgrounds; the lecturer with considerable knowledge of programming; the first year student with some knowledge of the same; the gamer, who may well have little knowledge of, or interest in, programming. We have written a questionnaire for our testers to fill out. This allows us to collect structured data which will be much easier to analyse than organic interview notes. It first establishes who the user is and what they know:

¡table¿

Next, we will ask if they could use the program to do the tasks it is designed for:

Could you perform the following tasks?

¡table¿

Finally, we ask if they could interpret the heap visualisation.

By channelling responses to each question into categories we make our analysis of results straightforward. We can easily differentiate between an unclear interface and lack of understanding in the user by separating results based on the first section of questions. If someone who has never used a debugger has trouble setting a breakpoint that is to be expected. If an experienced programmer cannot figure it out we have a usability problem. Bar charts would be good for representing this data?

The disadvantage of this approach is rigidity in response format. We can encourage testers to write down any opinions they have that the questions we pose have not allowed them to express. When reviewing each feedback sheet we take note of these comments. We plan to perform multiple rounds of user evaluation using this questionnaire. This will tell us whether the changes we have made have been effective in improving the system. By cross referencing improvements in results with actions taken over that period we can identify the changes that have been most beneficial to users. Ali wants to use the tool in his lectures. After the course, we could ask students who used it to answer the questionnaire, providing further data and allowing for more improvements. We can even have discussions with students while they are learning to use the product to learn more about their experiences.

5 Data Visualisation

5.1 Team Performance

To evaluate how our development team has functioned during the project we have performed 360-degree feedback, feedback that comes from members of an individuals immediate work circle. As we used Trello for project management we can use its Kanban flow to analyse our response time to adding new features

and ideas, this will allow us to know roughly how long it is taking us to produce a new feature once its been asked for.

We tried to maintain high performance in our team[2] by holding weekly team meetings, providing feedback in real time and negotiating the short-term and long-term goals of our project together. In our meetings we have regularly discussed the use of agile development as at the beginning of our project we agreed to complete weekly sprints to implement new features, however in practice we found that due to the varied work load of our other subjects this was an unreasonable ask.

For example, some weeks all team members would have multiple other pieces of coursework to complete, and other weeks many would have a much lighter work load. This meant that some weeks we managed to implement all the features we intended to and during other weeks the project was stalled while members finished other pieces of work, you can see this behaviour from our git commits graph below. To compensate for this we decided not continue using set sprint times but to carry on meeting regularly to report back on progress in each area of the project and to allow team members to raise any issues they had, for example, some members were unsure how the debugger was used and this issue was discussed and resolved in a project meeting.

We have elected not to analyse group performance by the number of lines of code written as this is not a useful metric on its own. Code Style Checking also proved to be unhelpful because it was introduced mid way through the project, resulting in a large number of unimportant warnings being raised about our reasonably consistent code style. Easy to integrate into our build process, but not helpful because everyone just ignored the checkstyle warnings. If we had prevented the checkstyle from halting the build it would have stalled the project by a number of days, wasting valuable time.

5.2 Burndown Chart

A burndown chart is a graphical representation of work left to do versus time, we split all our major features up into their component tasks and plotted them. As you can see from the graph we are close to being on track to the expected line, however we know from the previous GitHub commit graph that all our team members were busy somewhere between the 3rd November and 24th November, this matches with our data from our burndown graph as much less work was done around then.

5.3 Kanban Flow (Trello)

[5]

Initially we used planning poker to estimate how long each major feature would take to implement. Now using data from Trello we can map our estimates to the time between putting a todo note up and completing it on Trello, this is shown in the bar chart below.

As we can see from the chart we generally underestimated how long it would take us to implement a basic version of each feature in the project, this data has been useful as it made us acknowledge that our estimates may not always be accurate and we need to allow for slightly more time to complete any new features.

5.4 360-degree feedback

[3]

360-degree feedback is feedback that comes from members of an employee's immediate work circle, it includes direct feedback from our peers and supervisor, as well as self evaluation. Sometimes feedback from external sources such as customers is included but this would not be useful for our evaluation as we have not been working closely with an external client. We believe 360-degree feedback is more beneficial in our situation than traditional performance appraisal (where we would only be reviewed by our supervisor) based on the idea that anonymous feedback from multiple sources is more well-rounded

than direct feedback from a single source.

We used SurveyMonkey to ask our team to give feedback on how they believed the rest of the team to be performing. We asked about technical skills such as how they were performing in relation to the features they were implementing and how well they were working as part of a team.

The results graph shown is just for one persons feedback, with 1 being the lowest and 5 being the highest score they could have received in relation to that area.

The results from the survey highlighted the fact that while as a group we are on track for development, we could improve our communication in the group to avoid some members not understanding parts of the project or having to explain things twice when members arent in meetings. The feedback we gained from performing the evaluation will be used to plan the future development of our product as we are able to see where each persons strengths lie in the project.

References

[1] Alicja Nieborak, *Project planning: Evaluation plan*, http://www.jisc.ac.uk/fundingopportunities/projectmanagement/planning/evaluation.aspx

[2] Semantia, *How to Evaluate and Appraise Team Performance*, http://www.semantia.com.au/articles/productivity/how-to-evaluate-and-appraise-team-performance-in-virtual-teams/

[3] Jack Zenger and Joseph Folkman, *Getting 360 Degree Reviews Right*, http://blogs.hbr.org/2012/09/getting-360-degree-reviews-right/

[4] *Optimizelys explanation of AB-Testing*, https://www.optimizely.com/ab-testing

[5] *What is Kanban?*, http://www.everydaykanban.com/what-is-kanban/