

Diver User Study: Participant Summaries

April 26, 2011

Through analyzing audio/video recordings well as logged data about each participant's usage of the views in the Eclipse IDE, we were able to extract rich information about the ways that the participants approached the sessions. This document gives a summary of each users interaction with the Diver tool during each session.

1 Participant P1

Participant P1 began the first session by generating a trace and trying to analyse the software. Before attempting to gather a second trace, he attempted two keyword searches in his first trace, both of which failed. After spending approximately ten minutes familiarizing himself with the tool and the task at hand, he proceeded to attempt to solve the problem using software reconnaissance. After recording a second trace, he used the Program Traces View to select different threads, which updated the filter in the Package Explorer. He looked in the package explorer for meaningful words associated with the feature under investigation. After approximately 9 minutes, he found a method of interest and relied on the sequence diagram to analyse the feature, verifying his findings using source code.

For session **S2**, P1 used various methods to locate the feature, without much pattern. He tried browsing the sequence diagram as well as source code. He attempted two trace searches, both of which failed. In the end, he located his first foothold using mnemonic reasoning and by browsing the Package Explorer. His first foothold was found after 10.6 minutes from the capture of his execution trace

2 Participant P2

This participant began session **S1** by recording a single trace and using Diver's filter on the single trace. This showed in the Package Explorer the elements involved in the trace, but not unique to the feature. He noted that there is "far too much to actually look through," and so he recorded a second trace. Approximately 3.5 minutes after he completed his second trace, he was able to locate his first foothold in the Package Explorer. He verified his findings using

the trace search mechanism. From then, he made use of the Package Explorer to navigate into the sequence diagram and the sequence diagram to navigate to code. He used the sequence diagram and the source code editor equally for his analysis of the behavior of the software. He finished the session in short order, using the remaining time to investigate fine details of the feature such as the low level user interface functionality. He tended to verify his findings using the trace search facility.

In the second session, participant P2 began by anticipating the difficulty of the task stating, “So much happens in Eclipse,” and, “The filter was hugely beneficial.” After capturing the trace, he spent several moments exploring some sequence diagrams and noted, “Expanding the sequence diagram is not going to work.” He resorted to using Diver’s search facilities with mixed results. After 14 minutes and several failed searches, he found a first foothold. This behaviour differed from his first session in which he only used the search mechanism to verify his findings and not to discover new information. He also sighed numerous times during this session, a phenomenon which did not occur in the first session. After the first foothold was found, he followed his previous pattern using a combination of the sequence diagram and source code to analyse the feature.

3 Participant P3

Session **S1** was started by immediately generating two traces. Participant P3 began his investigation of the traces by using software reconnaissance and the Diver filters in the package explorer. He selected different threads in the Program Traces View and looked at the effect that they had on the Package Explorer. After locating a method of interest in the Package Explorer, he first investigated its source code and then revealed the first call to the method in the sequence diagram. He used the sequence diagram extensively to answer the first four understanding tasks, but relied on source code to answer the fifth. It took Approximately 7.6 minutes to find his first foothold.

Participant P3 had significant difficulty with session **S2**. He started with a very large trace which took Diver a long time to analyse. After a number of minutes waiting for the analysis, he became frustrated and attempted to create a smaller trace, noting, “No real world developer is going to wait ten minutes before trying to solve a problem.” Once the smaller trace was captured, he began exploration. He attempted to turn on the Diver filters, but was instructed that it was not allowed for the second session. He attempted to expand all activations in a sequence diagram which caused it to be very large. He stated that he did not like the sequence diagram for this session and abandoned it.

4 Participant P4

The participants were instructed to try and keep their traces as small as possible. P4, however, recorded extremely large traces during session **S1**. This

impaired the performance of the software reconnaissance filters during the session. After becoming frustrated with the tool’s performance, participant P4 stopped using the filters and resorted to using keyword searches in the trace. After several attempts, he was able to locate the feature for session **S1** in the sequence diagram and complete the session. He attempted the same strategy for session **S2** (using keyword searches in the traces), but was unsuccessful in this instance. He was not able to locate the feature and complete the session.

5 Participant P5

Participant 5 began with some confusion. He started session **S1** by gathering a single trace and trying to browse it. He was prompted to remember the instructions of the session and that he requires 2 traces and began again. His traces contained the information needed to complete the session, but he began to look for the wrong feature. After looking back at his instructions, he realized his mistake and was able to find his first foothold using the filtered Package Explorer after 5.6 minutes. Once it was found, he primarily used the sequence diagram to investigate the feature noting, it was “easy to see from the sequence diagram.”

To perform session **S2**, the P5 recorded a trace and began to explore it in the sequence diagram. After a little time, he decided to use Diver’s search facilities, but the search failed to return any results. searching to find where the functionality was located. He re-ran the program (without tracing) to ensure that he understood what his actions were that caused the feature to execute. He then browsed the package explorer to try and find classes that had meaningful names, and was not able to find any. He tried several more traces searches using Diver before he found one that yielded good results. After 10.3 minutes, he found his first foothold and started to answer the assigned questions. He relied mostly on the sequence diagram and his search history to complete the sessions, with some reference to source code.

6 Participant P6

Participant 6 began his first session with some time spent familiarising himself with the tool. After 5 minutes, he finished 2 traces and began software reconnaissance. He mentioned several times that he was tempted to do a code search, but refrained from it and used the package explorer and sequence diagram to find some foothold into the feature. He found it 3.5 minutes after capturing his second trace, though he said that he may have, “found that by accident,” but, “not totally.” From then, he attempted to answer the questions using the sequence diagram, but had difficulty understanding the visualization and got lost several times while trying to use the tooling. Though he was able to complete the questions mostly using the sequence diagram, he did not find it very intuitive.

For session **S2**, participant P6 captured a single trace and tried to activate his trace to use the filters, but was advised that it was not allowed for this session. He was quickly struck by the size of the sequence diagram, and noted that he was “really looking for a way to filter.” He tried to solve the problems using the sequence diagram, expanding many elements in it. He attempted to expand all of the elements under a particular activation, but was frustrated with how long it took and the size of the diagram after doing so. Contrary to session one, he avoided performing searches either in code or using Diver, stating several times that such searches would be “random.” In the end, he attempted several searches with one gaining him a first foothold after 37 minutes of investigation. He did not have enough time to answer any questions, however, and the session was incomplete.⁴

7 Participant P7

Participant P7 was a junior programmer with less than one year of experience. During his attempt at session **S1**, it was evident to the investigator observing the experiment that he had great difficulty understanding the software reconnaissance process and the design of Diver. Diver requires that each execution trace be captured within separate executions of the target application. Participant P7 consistently tried to capture everything within a single execution. In other words, he was unable to perform software reconnaissance because all of his captured traces included the feature under investigation as well as the features that were not a part of the session. Seeing that the participant was having extraordinary difficulty understanding the process, the investigator attempted to re-train the participant, but the participant was unable to complete the session within the time constraints. Having been given the extra training, he was able to complete session **S2**, however.

8 Participant P8

Participant P8 attempted software reconnaissance, but approached the problems differently than most participants. During **S1**, he started by gathering two traces and filtering based on them. However, he never used the Package Explorer to navigate to sequence diagrams. He navigated to source code with the Package Explorer, or he opened sequence diagrams using the Program Traces view. While browsing the Package Explorer, he never expanded the tree representation beyond the file level. During the interviews, he noted that this was a learned behaviour. He knew that the package explorer was filtered, but he, from past experience, still expected it to be very large so he tended to avoid it. He spent most of his time either reading source code or sequence diagrams. He was never able to find a foothold into the feature.

For **S2**, participant P8 gathered a single execution trace and primarily used a browsing strategy in the sequence diagrams. He was able to gain a first foothold

after approximately two minutes. He attributed this to learning effects. He said that during session **S1**, he spent a lot of time browsing sequence diagrams and learned that “Worker threads really are just workers.” That is, those threads with the name “Worker” were likely uninteresting, so he avoided them during his investigation and was able to narrow his search.

9 Participant P9

Participant P9 began his investigation by capturing two traces, and using a combination of the sequence diagram and source code. After several minutes, the investigator noticed a bug in the software, and the participant had to capture a new trace. After that third trace was captured, the participant filtered the package explorer to look through the sequence diagram and source code. He was very thorough and was even able to recognize when the feature’s functionality was spread across multiple threads (this feature executes in 2 threads, but for us to consider it completed, we only ever required the participants to find one of the threads). One point of confusion for him, though, was that he wasn’t always able to keep things in the source code editor synchronized with the sequence diagram. He expressed that he would like to navigate from the source code directly into the sequence diagram, rather than using the package explorer as intermediary.

He experienced much more difficulty in the second session. After some initial difficulty in the sequence diagram, he resorted to searching, but the searches failed. He then resorted to exclusively browsing source code. He expressed that he was aware that the session involved invoking an action from the context menu, and that Eclipse developers often use terms like “action” and “command” when implementing context menu items, so he started browsing for packages that had those words. He eventually found a class that he started reading through pages of source code. After some browsing of source code, he ran a trace search for some of the keywords that he had found in the source. Then he tried to understand the software using the sequence diagram. He expressed that what he was seeing in the sequence diagram didn’t quite match what he thought he understood from reading the code. He quit the session with before his time was exhausted, without finding a foothold or completing any of the understanding tasks.

10 Participant P10

Participant P10 started his investigation in session **S1** by launching the target application and familiarizing himself with the problem by trying various different interactions (opening a sequence diagram, swapping loops several times, etc.). He then captured his first trace in which he recorded a swap of a loop (he was investigating feature F2) as well as several mouse clicks, selections, etc. During his second trace, he tried several mouse clicks in the sequence diagram, but did

not exercise any functionality that would cause the sequence diagram to draw any figures. He performed software reconnaissance, by hiding the second trace. When he was unable to find a foothold after several minutes, he attempted several searches, all of which failed. He spent much of his time, then, trying to browse source code and sequence diagrams to find something of interest. Unable to find anything, he decided to start his investigation again by capturing two more traces. While investigating the results of those traces, he “hid” both of the traces from his previous investigation which, in fact, filtered out the software elements involved in feature F2. He was never able to find a foothold.

This participant was able to find a foothold for session **S2** in under eight minutes. To gather the trace for **S2**, he exercised the feature under investigation several times, pausing the trace for several seconds between each time he exercised the feature. He indicated that this was so that the timeline would contain “markers” that displayed where he exercised the function (Diver indicates that a trace was paused using a yellow marker in the timeline). After capturing his trace, he opened up a sequence diagram, and noticed that there was a blue marker in the timeline because he had a method selected in the package explorer. He right-clicked on the marker, to reveal what method was called at that marker. He then browsed the expanded sequence diagram and after several minutes found a class of interest which was his first foothold into the feature. He then proceeded to complete the understanding tasks using a combination of the sequence diagram, source code, and search.