Mylog: A Visualization For Mylar Log Data

Shawn Minto

The University Of British Columbia

ABSTRACT

Software systems use text based log files for many different applications, such as monitoring events, or in the case of a research setting, capturing the usage data from a user study. The downfall with these text based log files is that even if they are well structured (using XML or the similar), they are difficult to read and even more difficult to get meaningful information from. To help extract information, scripts are normally written, but this can be a difficult task. Also, the output from these scripts is normally text based and therefore they can only continue provide limited information. Mylog solves these problems by visualizing log data and presenting it to the users graphically. Mylog reduces the time to analyze the log files since it produces instant feedback on parameter choices and the user is able to have an increased confidence that the visualization returned the information that they were looking for. Also, Mylog allows further inspection after the data in question has been found.

1 INTRODUCTION

Programmers often utilize Integrated Development Environments (IDE) to search for and analyze places of interest in a programming project. The shortcoming of IDE's is that they present the entire structure of a system, which can be a burden to developers performing a simple task since they are concerned primarily with a small portion of the code. A popular Java programming IDE is Eclipse[1] from IBM[2]. A tool to assist developers with the dilemma of information overload is called Mylar[3]. Mylar is an Eclipse plug-in that helps users focus on their current task by only showing the structure that is relevant to them currently [4]. Mylar was developed in the Software Practices Lab (SPL[4]) at the University of British Columbia by Mik Kersten and Gail Murphy. Mylar has since been released on Eclipse.org as a technology project and is freely available for download.

As part of Kersten's PhD thesis on Mylar, a user study was conducted during the summer of 2005. This study involved over 75 real developers from industry using Eclipse with and without Mylar installed. Over the course of four months, the developers' usage of Mylar and Eclipse was monitored and collected for further analysis. The data collected is in an unstructured extensible markup language (XML[5]) format and contains rich information about the developers' usage. The different types of events that were logged are selections, edits, command invocations and preference changes. Each of these events includes: the type of event, start and end date stamps, the origin of the event, a string representation of the event, how the event affected the model in Mylar, and the object that event acted upon. Figure 1 shows an example of an element in the XML log file. For privacy considerations, the object that the event involved was obfuscated, but this data can still be useful when looking at the events since the object provides context for the event in question. Since many of these events are generated by a key-stroke or a mouse-click, large quantities of data are produced and it is difficult to visually inspect these large log files in text format; therefore, a visualization of this data would be beneficial for analysis purposes. This log data is not only beneficial for the purpose of the Mylar user study, but it provides general usage data for Eclipse which has yet to be explored and could be used to enhance Eclipse's usability.

```
<interactionEvent>
      <kind>selection</kind>
      <date>2005-10-31 18:40:07.811 PST</date>
      <endDate>2005-10-31 18:40:07.811 PST</endDate>
      <originId>org.eclipse.pde.ui.siteEditor</originId>
      <structureKind>?: class org.eclipse.pde.internal.ui.editor.s
      <structureHandle>?</structureHandle>
      <navigation>null</navigation>
      <delta></delta>
      <interestContribution>0.0</interestContribution>
</interactionEvent>
```

Figure1: An interaction event in XML

These events are stored in a structure called an interaction history. This interaction history is used in two different ways. The first is to monitor the usage of Mylar and Eclipse during the user study. This information allows the Mylar developers to determine the usefulness of their tool, as well as gain an insight into how Eclipse is used. This is the data that will be used for the visualization of Mylar log data. The interaction history is also used by the Mylar tool itself. In this second form, an interaction history is associated with each task that the developer defines. When a task is active, the events that are generated by a user are logged, as well as applied to the Mylar model. Using these events, Mylar is able to track what the user is doing and therefore limit the display of the structure to items related to the current task. When a task is deactivated, this interaction history is saved. This means that if a user reactivates a task, Mylar is able to restore the state of its model to help users remember important information related to the task and allow them to continue from the same point that they had left it.

The advantages of visualizations are that users have the ability to quickly identify trends and anomalies in data and easily analyze large quantities of data creating an increase in productivity. Visualizations can be used to help represent complex data to assist in building new knowledge. Since Mylar produces such large amounts of data, it would be more appropriate to explore this data visually instead of through inspection of the text files. Currently, custom scripts are being written to attempt to analyze the gathered log data to determine sequences of events, but this is an unacceptable form of analysis since it is unable to provide sufficient detail. Furthermore, a parser and report generator currently exist for this data, but only produce high-level overviews of the usage and not low-level details that would allow the exploration of single events and sequences of events.

[1] http://www.eclipse.org
[2] http://www.ibm.com
[3] http://www.eclipse.org/mylar
[4] http://www.cs.ubc.ca/labs/spl
[5] http://www.w3.org/XML

Currently, some difficult tasks are being performed with the assistance of custom scripts which provide minimal information to the user. In a perfect system, it would be ideal if a user could: easily search for a specific sequence of events, display the usage of Mylar and filter events to show the frequency of use of a single command (or sequence of commands) or to remove events that are not required for the current analysis.

Mylog is a tool developed to visually present and allow analysis of log files collected from the Mylar user study. Mylog presents the log file through the use of a sequential layout to maintain temporal positioning. This tool would be useful for someone who is analyzing the data collected from the user study to help them determine the frequency of sequences of events or commands. Furthermore, since the Mylar user study data also provides detailed usage information for Eclipse, Mylog would be useful for exploring how "real" developers use Eclipse. A final implementation of Mylog would ideally support the ability to compare multiple log files at one time. This was deemed to be fairly simple once one log file is visualized, and is therefore beyond the scope of this paper.

The remainder of this paper provides further detail regarding Mylog and its implementation. Section 2 discusses the related work pertaining to the visualization of log data. In Section 3, an overview of Mylog and its use of information visualization techniques are given. Section 4 briefly discusses the implementation of Mylog and Section 5 outlines some scenarios that Mylog can be used in. Following this, Section 6 is a discussion on the strengths and weaknesses of Mylog that were uncovered during development and testing of the system. Finally, Section 7 discusses some areas of future work to expand the abilities of Mylog and then the paper concludes.

2 Related Work

The visualization of log data is a limited area of research. The main areas of interest involve the visualization of internet and system log data. Both of these topics rely on actual dates and times to produce their visualizations, therefore putting them in the category of time-series visualizations. In addition, these areas do not normally deal with low-level events such as keystrokes, but higher-level events such as alerts and errors in a system, which happen much more infrequently.

Gray et al. proposed a solution to low-level interaction data with a system by providing a calendar based overview [3]. This method was the original inspiration for Mylog, but it was deemed that the time that the event occurred was not important for the purposes of Mylog. The calendar visualization allows for a quick overview of the usage of a system based on time [3]. Also, Gray proposed a method of displaying the usage of a system by producing a visualization on top of the system in question [3]. This idea could not easily be used for Eclipse and Mylar due to the extensibility of Eclipse. Since there are a huge number of plug-ins that a user could have installed, it would be very difficult to attempt this approach.

Another approach to log visualization is the SeeSoft view proposed by Eick et al [2]. Generally, SeeSoft views are used to represent data based on the time of their occurrence providing an overview of what events occurred at what times. This SeeSoft view is the inspiration for the final Mylog implementation.

Finally, in parallel with the development of Mylog, another project by Karen Parker was being developed to visualize privacy information regarding to web browsing [6]. Parker's visualization continues to use time as a variable for the layout of the events; however, in the final presentation of the project, a compressed view was displayed that was similar to Mylog. The difference with Mylog is that Parker's visualization places all of the events, regardless of type, into a single line. This is reminiscent of a SeeSoft view and requires the users to identify different events through the use of color. Also, Parker's visualization requires a user to scroll if the data is unable to fit onto the screen, whereas this is not the case with Mylog.

Mylog differs from current research since the sequence of the events is the key information not the time of the event like the current log visualizations. This means that the data cannot be easily aggregated in dates or times since it can obscure the real information that an analyst is looking for. Also, Mylog needs to visualize a large number of data points that are collected due to low-level events such as key-strokes. Due to these differences, Mylog was unable to make extensive use of previous research in the area of log data visualization; therefore, Mylog uses techniques that are designed for large data sets such as bifocal lenses and filtering, as well as draws some of the layout techniques from SeeSoft views.

3 Mylog

Mylog eases the analysis of Mylar log data through the use of a visual representation of the log while providing tools so analysts can gain further knowledge of Mylar and Eclipse usage. The visualization is created simply by invoking an action within Eclipse and selecting a set of log files that should be visualized. Mylog analyzes this log data and provides a visual representation of it in a separate window as shown in figure 11.

Mylog presents the data in a linear fashion as to retain the sequential nature of the events in the log file. Since events are only related based on their sequence in the file, no arcs are required in the visualization. This also means that there are no problems with occlusion that needed to be dealt with during the implementation of this tool. Each node in the visualization represents either a single event or a group of similar events as determined by the aggregation algorithm. To make an effective visualization of this log data, Mylog employs many different information visualization techniques that are discussed in more detail below.

3.1 Color

Mylog uses color so that analysts are easily able to determine the kind of event that each of the rows in the visualization is showing. The following is a mapping of event to color:

- Command = Gray
- Selection = Blue
- Preference = Cyan
- Edit = Pink

By using color to distinguish the event types, labels for each row of events did not need to be provided and therefore the screen real-estate could be effectively used for the visualization. This proved to be beneficial since text is know to consume valuable real-estate quickly. To provide this mapping of event color to

event, a legend is provided to the user in the bottom right of the display for quick reference as can be seen in figure 2.

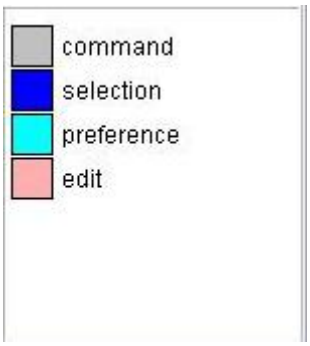

Figure 2: Mylog legend

To avoid overwhelming the analyst with excessive amounts of color, nodes are not normally outlined in a different color. Since the nodes are not outlined, there is no visual separation of the nodes unless one is brought into focus. When a node is in focus, its color changes so that the analyst is able to easily determine what node the details are related to. Mylog supports two different types of focus: hovering over a node and clicking on a node. When a node is hovered over with the mouse, it is colored pastel orange, and when it is left clicked to bring it into focus, it is colored black. This means that if a node is colored black, the linked details view contains information about that node.

Also, Color is used to highlight the nodes that belong in a sequence that was specified by the analyst. When a sequence is specified by the analyst, that nodes matching this sequence are highlighted using a red outline. Since color can be preattentively perceived, the highlighting of sequences in this manner allows the analysts to quickly perceive of the location and frequency of a sequence of events in a log.

3.2 Aggregation and Spatial Layout

To make an effective visualization of a large number of nodes, the use of aggregation as well as a good spatial layout is needed. It was originally decided that since the data had timestamps, it could be aggregated and displayed using this information. This would have meant that the visualization would be required to display a smaller number of nodes since the data could be aggregated per day. This was determined to be incorrect since the time is unimportant and is only used to provide a correct sequencing of the events.

Mylog aggregates the events to be visualized by combining similar events into a single node in the sequence. The similarity of the node is calculated based on all of the attributes for the event other than the times. This means that the type of event, object acted on, origin of the event and the string representation of the event must all be the same and the events must occur sequentially. Each visualized node adds an extra attribute over the original event that contains a count of how many events it represents. It has been found that this can reduce the number of nodes displayed by about 25% in a large log. Even though a node may represent multiple events, the nodes size remains constant. Keeping the node sizes the same was decided upon since aggregation is used to reduce the size of the visualization, allowing all of the nodes to be displayed without making them too small. It would be useful if there was an easy way to represent the number of events that each node is representing. This was not implemented, and it is thought that a color gradient (light to dark) for each of the event types could be used to provide this additional information to the analyst.

Determining the spatial layout of the nodes in Mylog was a simple task. The only difficulty in the layout of the nodes was determining the size that each node should be so that the entire log will fit within the screen bounds. The log data and tasks dictated that the nodes be laid out in a sequential manner, so the naïve method would be to put all of the nodes onto a single line like the SeeSoft view [2]. This method works well with the use of color to distinguish the nodes, but it becomes more difficult to inspect since all of the data is clustered on a single line. To avoid this, each event type was placed in a different row, but the nodes were still placed sequentially in the horizontal direction (figure 11). This allows analysts to easily determine how frequently users jump between different types of events, as well as quickly identify outliers in a log file. Also, providing this separation allows the analysts to quickly identify areas of interest if only a certain event type is of important to them. Since all of the nodes are shown sequentially, there is no node occlusion as previously mentioned. Conversely, since they are laid directly one after another, there is no definitive separation between each of the nodes. This could be solved by outlining each of the nodes, but as discussed earlier, it causes visual clutter especially when the nodes are tiny and therefore was not implemented.

3.3 Focus + Context

In many large visualizations, users need and want to see more details about a certain area of the visualization while still maintaining an idea of the overall context of the area that they are viewing. Some approaches to solve this include bifocal lenses, fisheye lenses and separate overview and detail windows. Since Mylog is visualizing massive amounts of data, it is necessary that the analyst can view details in a section of data while still maintaining the global context of their position in the log file, so a good focus + context model is needed.

Mylog uses a bifocal lens to allow users to see detailed information about a subset of the log while still allowing them to maintain awareness of what area of the log they are in and what events are happening around it. This bifocal lens is only implemented to be unifocal. This means that the vertical direction of the zoomed portion of the visualization remains constant while the horizontal portion changes as the user moves their mouse horizontally as can be seen in figure 12. The lens was implemented in this manner since the analysts need to be able to view a node of interest as well as other nodes in its vicinity. By having the vertical axis remain constant, all nodes in the vicinity of the node in question are the same size allowing for further inspection. Also, to ensure that users are able to leave the visualization while maintaining their current context, the right mouse button is used as a modifier to enable the movement of the lens. This means that if an analyst is inspecting the data, then decides to add a filter or a sequence, their current view will not change when the mouse is moved.

3.4 Linked Views

Linked views are useful when visualizing data so that more detailed or other related information can be provided to the user.

Mylog takes advantage of linked views in two ways: a visualization information section and a node detail section. Both views are positioned directly underneath of the visualization as can be seen in figure 12. First, a small section in the bottom center of the user interface (UI) is used to display information directly related to the visualization and its current state. In this section, information, such as, the total number of events and the number of nodes drawn is displayed to allow the analyst to know how the aggregation worked. Also, this section shows how many nodes are currently filtered and how many sequences have been found in the visualization (figure 3). This data allows the analyst to get a quick overview and feedback of their actions involving the filters and the sequence highlighting.

Figure 3: Visualization detail view

Another use of linked views in Mylog is for displaying the detail of a node. This is very useful since an analyst needs to be able to view at all of the information that a node represents to make an informed judgment of what the potential sequences are and how useful a node is. When a node is in focus, the details of the event that it represents are displayed in a text box located in the bottom left of the UI. A text box was used so users are able to copy details out of it so that they can place them into either a sequence or a filter. This view displays all of the information that is contained in the event(s) that the selected node represents, as well as the number of similar events that are aggregated into that node (figure 4).

Figure 4: Node detail view

To provide extra detail, tooltips were going to be used when a node was hovered over with the mouse. This tooltip would have displayed all of the same information as the linked view, the detail provided from the XML. This feature was not fully implemented due to time constraints. This should have been a simple feature to add, but the tooltip continuously disappeared when the visualization was refreshed. The Mylog visualization needed to be refreshed to reflect the current coloring and filtering of the nodes. If more time was available, the refresh of the visualization could have been modified to accommodate the use of tooltips by only refreshing when required and displaying the tooltips after the refresh had completed.

3.5 Filtering and Highlighting

Filtering is used to help users of a visualization manage its complexity by removing objects that are not important to them from the display. On the other hand, highlighting is used to help users identify their current location and points of interest in the visualization. Mylog employs both of these techniques to allow for extensive control over the data set that is being visualized assisting in the discovery of event sequences and other information.

First, Mylog has an extensive filtering mechanism to allow analysts to reduce the clutter of the visualization. This reduction can either be done using an inclusion or exclusion filter. The inclusion filter shows only the elements that meet the filters criteria. This means that an analyst can pinpoint events of interest and determine their frequency of use. Conversely, the exclusion filter hides nodes that match the filter criteria; therefore, reducing the clutter by removing unimportant nodes. The filter criterion is specified on a per attribute basis therefore making it very powerful. This filter information is specified by the analysts through a dialog box as shown in figure 5. The information supplied to the filter is used to either hide or only show the nodes that match. Matching is done based on whether the attribute of the node in question contains the specified value. This substring based matching makes a powerful filter mechanism.

Figure 5: Add filter dialog

Highlighting in Mylog is used in two different ways. The first orients analysts to their current focus point. The details for the currently selected node are displayed in a separate window and a connection between these need to be made. A selected node is determined by the user expressing interest by left-clicking on the node. This action will enable the linked views to be updated accordingly. Highlighting is also used for the displaying the current node that the mouse cursor is positioned on. This allows users to determine the node boundaries since they are not explicitly defined. Next, highlighting is used in Mylog to display nodes that lie in the current sequence that is specified. The

sequence is specified using the same mechanism as the filter data is specified. This highlighting allows users to quickly determine the frequency and locations of the specified sequence (figure 6 and figure 13). The colors used for highlighting have been mentioned in Section 3.1.Since the order of the sequence matters, the sequence list has support for moving the sequence items up and down the list to quickly change the order of the events.

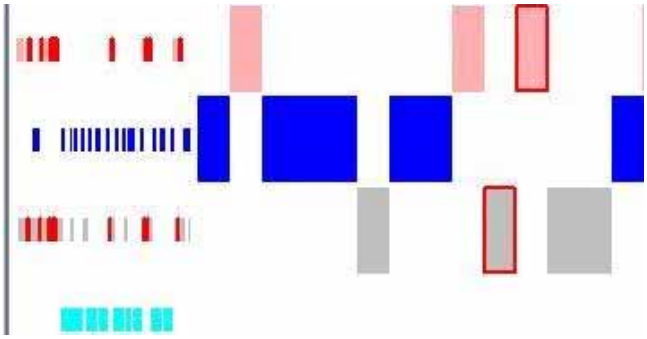

Figure 6: Sequence highlighting

4 IMPLEMENTATION

Mylog was implemented using Java 5.0 as a single plug-in for Eclipse. This allows Mylog to use the existing parsers and tools for the log data that have been implemented in other Mylar plug-ins. For the visualization, Mylog makes use of the Prefuse toolkit[6] which works with Java and Swing. Prefuse was chosen because it is a simple to use toolkit that works well with Java. Prefuse is easy to learn, as well as easily extensible, allowing for customizations to be added to support the Mylog visualization. Also, Prefuse is a highly functional toolkit providing many predefined standard information visualization techniques that can be used without modification. Mylog's user interface was built using Swing, which is an application programming interface (API) for building UI's that is distributed with most Java installations. Both of these toolkits were chosen for their ease of use to aid in the quick development of Mylog due to the strict schedule of this project.

Mylog extended only a small number of classes from Prefuse, allowing for customization of how they behave. The two major changes were to the layout and to the bifocal lens. A new layout was created to position the nodes sequentially while taking into account the size of the screen to adjust the node size, ensuring that all of the nodes can be visible at one time. The bifocal lens had to be changed so that the magnification and range of the lens was dependant on the number of nodes that were being displayed. Originally, the lens parameters were static, however, it was discovered that it was not useful since it either made the nodes too large, or did not magnify at all. The functions that were required for the lens were determined through performing trials for many inputs. These trial results were then entered into Microsoft Excel and graphed to determine the function that should be used. It was found that the magnification function was an increasing polynomial function based on the number of nodes on the screen, and the range of the magnification was a decreasing polynomial function of the magnification. Also, a filter based on attributes, the way of determining the color of the node and the focus control were modified to provide the support that Mylog required. The functionality for determining sequences was implemented independently of Prefuse, ensuring that the data is processed sequentially. Once the sequences were determined, the nodes that were part of a sequence had an attribute set to inform the visualization that the node should have a border that is a different color than the node.

Originally, Mylog was to be developed strictly as a plug-in for Eclipse since the rest of Mylar and its support tools would be available. This turned out to be difficult to implement since the Prefuse uses Swing and Eclipse employs a different library called the standard widget toolkit (SWT). SWT has a way of integrating a Swing view into one of its own views, but due to refresh policies, excessive flickering of the visualization resulted. This was deemed to be distracting and the UI was extracted and redeveloped in Swing. To ensure that the existing tools could still be used, the UI for the visualization is launched from within Eclipse after the data has been imported.

5 RESULTS

Throughout the development of Mylog, some use scenarios were the driving force for the features that Mylog needed to provide. These scenarios are presented in the next section. Also, the performance of the system was informally measured on three levels and these results are reported in section 5.2.

5.1 Scenarios of Use

Mylog is a tool designed for use while analyzing log data from Mylar, however not all tasks would benefit from the use of this visualization. The following are three scenarios of use in which Mylog would help the Mylar log data analysts.

5.1.1 Frequency of Sequences

The scenario used for driving the development of Mylog was the ability to display sequences of commands to the analyst. Displaying these sequences would allow the analyst to quickly identify if they existed, as well as the frequency and number of the selected sequence. Currently, to determine sequences in Mylar usage data, custom scripts are written to parse the log files. The problem with using scripts is that they are difficult to write and even more difficult ensure that they are correct. These scripts consume much of the analysts' time and therefore decrease their productivity. Another drawback to using scripts is that they can only provide limited information, such as, the number of sequences, or even worse, another file containing all of the related sequences so that detailed information can be provided. Other problems with scripts are that small changes in a sequence require reprocessing all of the log data and a lack of contextual information for the sequences, such as, location in the file.

Sequences of events are important for the analysts of this data, as well as designers of Eclipse and Mylar. If certain commands are always or almost always performed together, it would be beneficial to either place them near each other or combine them into a single action. This redesign of the actions would benefit the developers using the system by reducing the number of commands that they must invoke to complete the task that they are interested in. An example of this is if the actions of

[6] http://prefuse.sourceforge.net

refactoring[7] a class to have a different name and moving the file to a new package were commonly performed together. If this were the case, it would be beneficial for Eclipse to support specifying a new location when the refactoring command is invoked on a file.

Mylog allows the analyst to enter a number of events that they are significant into a list that represents a sequence. This method allows the sequences to be preattentively processed through the use of colored highlighting as previously described in section 3.1. An additional benefit to the visualization of these sequences is that if the analyst modifies the sequence that they are interested in, instant feedback is provided. This instant feedback allows analysts to quickly determine if the sequence that they are concerned with exists, or if their modification was correct. Also, since the highlighting of a sequence does not hide nodes, the analysts are able to view the contextual information around them, such as, location in the log file and other interactions that exist around the sequence. The only downfall to the analysis of sequences in Mylog is that the analysts must still use predisposed knowledge to determine what sequences of interest.

5.1.2 Command Usage

Another excellent use of Mylog is to determine command usage. The usage of a command could include the number of times that it was used, the frequency that it occurs in the log or even the distribution of its use within the log (only used in clusters or evenly dispersed). Currently, grep[8] could be used to count the number of occurrences and even compute the frequency, but yet again, this is time consuming and does not allow for quick, preattentive display. Also, it is difficult to determine the distribution of the usage of the event in question. This can help with the analysis of the usefulness of the log file (as will be discussed in the next section) or whether specific functionality is utilized. This information would benefit a developer of Eclipse or Mylar because if a command is not used, it could be removed from the application since it is a dead feature. Also, if a command is always used in clusters, the developers could attempt to identify a way to easily invoke the action multiple times sequentially, therefore improving Eclipse or Mylar to assist their respective users.

Mylog easily supports this through the use of either the sequences or the inclusion filter. First, an analyst could enter a sequence of one event into Mylog. This would cause each of the events to be highlighted for further analysis. This method allows the analyst to maintain the context of the surrounding events while investigating the usage of the command in question. The sequence method would be most useful in determining how a command is used and what events are performed around it. In contrast, by creating an inclusion filter with the command in question, all other events are removed from the screen. This method quickly allows the analyst to discover the frequency of use and its distribution within the log file. This method reduces the visual clutter that is evident when creating a sequence; therefore, making it simpler for quick analysis.

[7] Refactoring is the process of modifying a class and all of its references together to maintain code consistency. This means that when a file or method is renamed, compilation errors are avoided and the program behaves the same.

[8] http://www.gnu.org/software/grep/grep.html

5.1.3 Usefulness of Logs for Further Analysis

While developing Mylog and analyzing log files for testing purposes, it was discovered that Mylog is well suited to assist in determining the usefulness of a log for further analysis. This measure of usefulness is needed since Mylar log data was not collected in a controlled setting like most user studies, but rather many "real" developers used it in their everyday programming situations. Since the study was not controlled, it is difficult to determine whether the information contained in the logs are useful for further analysis to determine the usability of Mylar. Currently, log files are discarded if there are not a certain number of events and if Mylar was not used sufficiently. These cutoff numbers are hard coded into the different report generators within Mylar. The current way of determining whether to keep or discard log files can make incorrect choices due to its lack of information; therefore, it would be greatly useful if there was a means of visually inspecting the files to determine if they suit the needs of the analyst, therefore producing better analytical results.

Mylog can be applied in many ways to determine the usefulness of a log file. The first is through a quick visual inspection of the file. During development, a log file was found that contained few edits or selections, but many command invocations. Normally, logs have a relatively even distribution of edits, selections and command invocations. It was found that the log was a developer debugging an application since all of the commands were related to this type of task. Even if this log passes all of the current tests of the number of events and command invocations, it would not be a useful log for analyzing if the analysts interests deal with editing a file. Also, it has been found through this process that some logs contain only commands, and sometimes only one type of command, next word. This was interesting to discovery since the number of events was used during the user study to determine if the user could be promoted to the next phase of the evaluation. This discovery exhibits that the user had found a way to become promoted earlier, allowing them to use the tool without providing the prerequisite usage data.

Finally, Mylog can be used for determining the usefulness of a log file is through the inspection of command usage. The scenario of command usage has already been discussed in the previous section, and will not be discussed further here. Currently, the Mylar analysts utilize the number of times a specific command is used to determine if a log is acceptable for further analysis. It has been determined that this may be incorrect since the command could occur many times in one location of the log, but the analysts expect it to be distributed, showing the proper use of Mylar. It is unknown how many log files this might affect, but it would allow for better data to be used when visualizing the results of the user study.

5.2 Performance

Mylog's performance was gauged by three measures; speed, its ability to support the required tasks, and the color scheme that was chosen. The performance of speed was based on two different factors: the interactivity of the system and the time to load the visualization. In general, Mylog maintains interactivity for a few thousand nodes which sounds acceptable, but in reality, log files can exceed 20,000 events which need to be visualized. With this many nodes, interactivity is lost and therefore it is difficult to use the system. Even though the interactivity is lost, Mylog is still able to allow an analyst to identity sequences and navigate the log

file data with only minor frustration. Next, the time to load the visualization was analyzed and it was found that this can also be slow for larger log files. The root of this problem is that the log data is parsed multiple times, once to get into the system, then again to aggregate the data and then again to create the graph. This load time is due to the use of a parser that currently exists for the data, therefore, if a custom parser was created to import the data, all of these steps could potentially be completed at once and the load time of the system would be greatly improved.

Next, the system was evaluated on its ability to perform the required tasks. Since the system was built with a set of common tasks in mind, it is able to support them well. An explanation of these tasks is provided in the previous sections. Also, it was found that the highlighting of sequences and filtering of nodes is virtually instantaneous, therefore allowing users to directly see the impact of their actions on the data set.

Finally, the color scheme was evaluated using VisCheck[9] which is an online tool to simulate three different forms of color blindness: Deuteranope (a red-green colorblindness), Protanope (another red-green colorblindness) and Tritanope (a blue-yellow colorblindness which is rare) [1]. Since 1 in 20 people lack the ability to see all colors [1], it is very important to design for these people when creating a visualization that distinguishes information using color. Screenshots of the color scheme were taken and uploaded to VisCheck to create images that would be similar to what a person with each of these forms of colorblindness would see. It was found that the color scheme was chosen is favorable for people with any of the three deficits that were checked. The results of VisCheck can be seen in figures 7 - 10.

6 DISCUSSION

Through the development of Mylog, many lessons were learned about producing visualizations of large data sets. These lessons will be discussed further in the next section. Also, Mylog has both strengths and weaknesses. Some of them are based on the lack of time to implement them, while others such as performance are limited by the technology used to create the system. Mylog has been used to analyze many different log files throughout its development. These files have ranged in size from small (1000 events) to a large ones (over 20,000 events).

6.1 Lessons Learned

While developing Mylog, several things were discovered about creating visualizations:

- When creating a visualization that has real users, many techniques from human computer interaction should be followed such as gathering task examples early and involving users in the design of prototypes. This was evident since the tasks were collected early, but the prototype was designed without user input and therefore had to be changed.
- Visualizing sequence data is difficult since there are very few techniques that can be used to assist with linear data.
- Determining how to aggregate data is difficult.
- It is difficult, if not impossible, to be able to display tens of thousands of nodes while sill maintaining interactivity when using a toolkit with Java. Therefore, I now see the need for developing visualizations using real computer graphics engines and techniques.
- Color schemes are difficult to determine especially when items are highlighted and therefore two colors must interact well to draw a user's attention, while still supporting color blindness.
- When there are multiple attributes that can help specify a node, it is difficult to determine structures that allow for quick access to the nodes, while maintaining low memory usage.
- Making a visualization portable is an ambitious problem since much of it depends on both the UI toolkit as well as the visualization toolkit used.
- If a user requests a visualization of some data, there is no guarantee that it will actually get used.

6.2 Strengths and Weaknesses

Mylog is a prototype visualization tool for Mylar log data and provides excellent support for viewing an overview of a user's interaction, as well as quickly identifying locations where a specified sequence has occurred. Since Mylog is a prototype, it does have some weaknesses, such as, the potential for clutter due to the large number of nodes being displayed, and there is no way of automatically determining sequences in the data. The following two sections will discuss these strengths and weaknesses further.

6.2.1 Weaknesses

Mylog is not a perfect tool, and has some weaknesses associated with it. The two major weaknesses are the potential for screen clutter and the limited use of humans' perceptual abilities. Also, Performance is a weakness of the system but it has already been discussed in Section 5.2 so there will be no more detail on it in this section.

The first limitation of Mylog is that there can be excessive screen clutter. This is directly related to the data that is being visualized along with the aggregation support that is implemented. Since there is a large amount of linear data, it is difficult to place it on the screen in a meaningful manner while maintaining a clean UI. This could be solved by using a different aggregation technique or determining a better way of displaying the nodes in a meaningful manner. In this project, a naive aggregation was used, and it was found to reduce the number of nodes displayed by up to 25%, but a better aggregation technique would reduce the number further.

The second major drawback is that there is limited use of a human's ability to preattentively process information. Currently, the nodes are laid out based on their location in the sequence of events and the type of event that occurred. This means that the analyst can only identify outliers or anomalies in the log (such as

[9] http://www.vischeck.com

one type of event occurring infrequently) quickly. In addition, if a sequence is input, users are quickly able to determine the locations and frequency of it, but this requires knowledge of what they are looking for. To exploit this preattentive ability directly on startup of the visualization, it would be beneficial to preprocess the data using heuristics to attempt to determine common sequences of events.

Finally, many tasks, especially when looking for sequences, require knowledge of the surrounding nodes which is not supported. Currently, Mylog only supports the selection of a single event for inspection, meaning that if multiple nodes are to be examined, they must be selected one at a time and their data either be remembered or saved in an external location for later use. A better approach would be to allow the selection of multiple events in a range and display their details together so that the analyst would be able to gain further knowledge of the sequence that they are looking at.

6.2.2 Strengths

Even though Mylog has some limitations as outlined in the previous section, it supports the common tasks that are currently done through the use of custom scripts and visual inspection of the log files. Since Mylog visualizes this data, it is easier to inspect the files as well as view sequences of data along with their location in the overall structure of the log. Mylog has three main strengths: the ability to quickly identify specified sequences, the ability to view the entire log file, and a powerful filter and sequence search.

The first main strength is that once a user specifies a sequence that they are interested in, they are quickly able to determine its locations, as well as, the number of them. The location and the general frequency of a sequence is easily picked out by users due to preattentive processing through the use of highlighting. This means that users are able to quickly determine the location of a selected sequence without doing extensive inspection since these sequences are highlighted.

The second asset of Mylog is that it is able to display an entire log file to a user while still maintaining the ability to gain detail about a single event. This is a great improvement over the current method of inspecting the log files since only a small subset of the information can be displayed and it is difficult to get an idea of what the overall structure of the log is. Mylog provides an overview of the entire log file allowing the analyst to quickly identify outliers of events or anomalies. One such anomaly discovered is that a log file that was visualized did not have any edits or selections for the most of the log. This would have been very difficult to determine from the log file directly, and therefore the analyst is able to infer what the user was doing, or if there was a potential problem with Mylar. Finally, an overview is not useful on its own since the events contained in the log are of real importance. Mylog allows users to gain further detail about these events. First of all, Mylog provides a bifocal lens to allow zooming into a group of nodes for closer inspection of the patterns. Along with this, a user is able to select a node and the detail that is provided in the log is displayed so that they can further understand the usage of the system.

The final core strength with Mylog is its powerful filtering and sequence capabilities. Both of these are needed to reduce the complexity of the visualization and highlight important data that an analyst is interested in. Both the filter and sequence mechanisms allow users to specify substrings of the data that they are interested in for any of the fields in the log. This means that a user is able to create complicated filters and sequences to gain a better understanding. Furthermore, since the filter supports either inclusion or exclusion, an analyst is able to simplify their display by eliminating events that are not important to them or only displaying events that they care about.

Furthermore, it is very simple to create a visualization using Mylog. To do this, a user only needs to invoke an action in Eclipse, and then select the log files that they wish to process. Files from multiple users can be selected since the filename specifies the user that the file is for. From here, the log files are automatically parsed and a separate window is opened to display the visualization. In this window, the analyst is able to choose the user that they are interested in and that log file is visualized.

7 Future work

To make Mylog a more effective tool for the analysis of Mylar log files, there are several items that should be implemented. One of the most important items would be to integrate the visualization from Mylog into the Eclipse IDE. This would allow the users analyzing the data to work in one window; therefore, allowing the integration of their tasks into a single environment. As discussed earlier, this was originally attempted by putting the Swing-based visualization from Prefuse into an SWT view, but it caused flickering which was distracting and unavoidable. To solve this, a visualization tool like Zest that is being developed at the University of Victoria could be used since it is developed with SWT. Zest was unable to be used for this project since it is still under development and therefore common visualization techniques like bifocal lenses are not yet supported.

In addition, the visualization of the data could be improved by determining a better way to aggregate events. Currently, a series of events is aggregated to a single node if they are the same event occurring directly one after another. This could be solved by allowing multiple event types to be aggregated together, such as edits and selections into one node since it was found that selections and edits of the same element occur frequently.

Mylog should support viewing multiple log files in a final implementation. This would be useful so that different logs can be compared allowing similarities between them to be determined. Supporting the viewing of multiple log files should be uncomplicated since the single visualization has already been produced, but the challenge is how to maximize screen real-estate for each log so that the details can still be gained. It would be beneficial in this stage to use heuristics to determine similarities in the log files and display these similarities in a way like TreeJuxtaposer [5] presented them. This would allow for quick comparisons between the log files and the ability to determine common sequences of events. It would also be beneficial to do further preprocessing on the log data to automatically determine potentially interesting sequences of events. This would build the knowledge of the analyst so that they are better able to determine the usage of Mylar and Eclipse.

Another item that would be useful is the ability to rearrange the order of the levels that each of the events are displayed on. This would be useful since different types of sequences or information would be able too be determined easier depending on the special

locality of the different types of events. Currently, edit and selection events are placed next to each other since they occur together frequently, but this may not be the best way to lay out the events when looking at certain sequences.

Finally, Mylog should allow users to specify multiple sequences of interest that are highlighted using different colors. This would allow an analyst to determine the relationship between different sequences of events. Also, this would allow for the persistent highlighting of important events which currently is not supported since only one sequence is able to be displayed. To further support this, it would be advantageous if the user is able to manipulate the color scheme to suit their needs. This would mean that people with color blindness would be able to choose colors that support their needs, and users that are not would be able to select colors to highlight information that is important to them.

8 CONCLUSION

Mylog is a tool that was developed to assist with the analysis of log file data from the Mylar user study. This tool is partially integrated in Eclipse and is aimed for people who wish to analyze the data that has been collected either on Mylar or Eclipse from this study. Mylog uses the power of visual representations to reduce the frustration that is incumbent when attempting to visually inspect log files and write scripts to extract information from them. Mylog uses many information visualization techniques such as focus + context, filtering, color and special layout to help analysts quickly find the information that they are interested in. Three scenarios were provided to show the usefulness of Mylog and how it reduces the complexity of analyzing the log files manually. Mylog still requires some work to be extensively used for analyzing Mylar log data, but even in its current state should be a powerful tool for some analysis tasks.

ACKNOWLEDGEMENTS

Gail Murphy for input throughout the project about the functionality required for analysts of the log data. Kenedee Ludwar proof read this paper before the final submission as a sanity and grammar check.

REFERENCES

[1] H. Brettel, "VisCheck", *www.vischeck.com*, Stanford University, 2003.

[2] S. G. Eick, et al, "Graphical Analysis of Computer Log Files", *Communications of the ACM*, Vol. 37, No. 12, 1994, pp. 50-56.

[3] M. Gray, et al, "Visualizing Usability Log Data", *IEEE Symposium on Information Visualization (INFOVIS '96)*, 1996, pp. 93-98.

[4] M. Kersten, "Mylar Technology Project", *www.eclipse.org/mylar*, 2005.

[5] T. Munzner, et al, "TreeJuxtaposer: Scalable Tree Comparison using Focus+Context with Guaranteed Visibility", *ACM Transactions on Graphics*, Vol. 22, No. 3, pp. 453-462.

[6] K. Parker, "Web Browsing Log Files Analysis", *http://www.cs.ubc.ca/~tmm/courses/cpsc533c-05-fall/projects/parker/proposal.html*, 2005

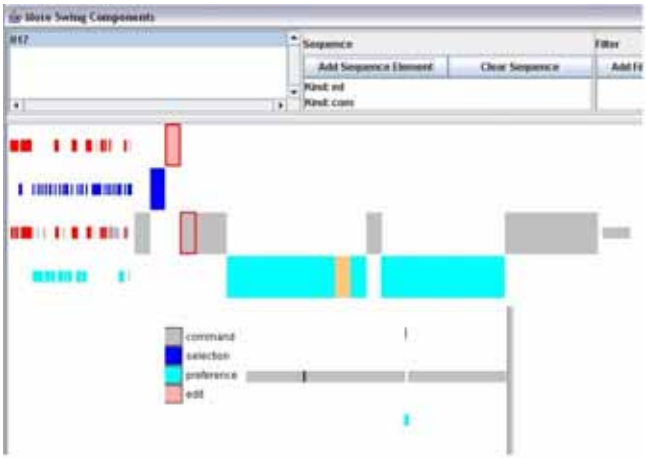

Figure7: Original Mylog color scheme.

Figure 8: Protanope colorblind color scheme.

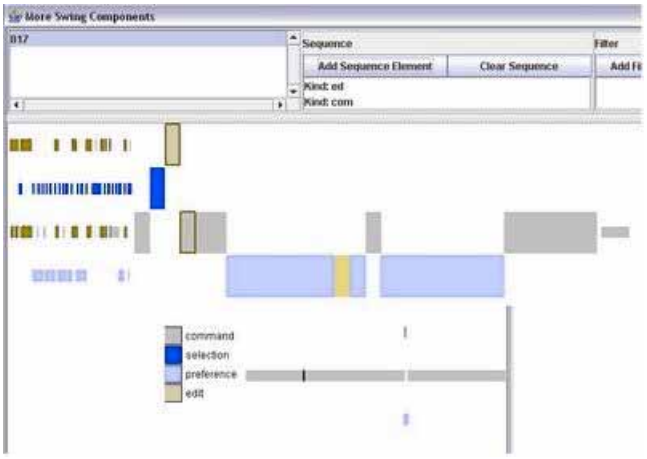

Figure 9: Deuteranope colorblind color scheme.

Figure 10: Tritanope colorblind color scheme.

Figure 11: Mylog

Figure 12: Unifocal lens

Figure 13: Sequence highlighting