

# ULRR

## Eclipse plug-in to monitor the programmer behaviour

Item Type	Meetings and Proceedings
Authors	McKeogh, John;Exton, Chris
Citation	eclipse '04 Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange; pp 93-97
Publisher	Association for Computing Machinery
Download date	2026-06-06 16:51:57
Item License	<a href="https://creativecommons.org/licenses/by-nc-sa/1.0/">https://creativecommons.org/licenses/by-nc-sa/1.0/</a>
Link to Item	<a href="https://hdl.handle.net/10344/2234">https://hdl.handle.net/10344/2234</a>

# Eclipse Plug-in to monitor the Programmer Behaviour

John McKeogh,  
Symantec,  
Dublin,  
Ireland.  
[John\\_mckeogh@symantec.com](mailto:John_mckeogh@symantec.com)

Dr. Chris Exton.  
Department of Computer Science and  
Information Systems.  
University Of Limerick,  
Limerick,  
Ireland.  
[Chris.exton@ul.ie](mailto:Chris.exton@ul.ie)

## Abstract

Comprehending and maintaining software is one of the core software engineering activities from early implementation to long-term software evolution. This paper describes an Eclipse based tool, which provides some quantitative insights into how different programmers develop and maintain software. In addition it presents a comparative pilot study that utilised the tool on a number of student programmers to gain insights in to how they utilise an interactive Development Environment.

## 1. Introduction

One of the most challenging aspects of research on programmer behaviour is the creation of suitable research methods and sources of information that facilitate the comparison and evaluation of different tools and techniques [1][4]. This plug-in tracks a number of significant behaviours such as mouse movement, key strokes (including paging), scrollbar movement and document changes that take place during the session.

Unlike some other existing tools, which have been developed, solely to monitor programmer behaviour [3] the tool we have created

*OOPSLA'04 Eclipse Technology eXchange (ETX) Workshop*,  
Oct. 24-28, 2004, Vancouver, British Columbia, Canada.  
Copyright 2004 ACM

accomplishes this via an ecologically valid manner. As Eclipse [2] is a mainstream well understood and complete development environment. Studies carried out using the tool are within an ordinary open editor environment in Eclipse. The tool tracks and stores the behavioural information in an xml format. This information is then processed with Xml2Excel, a tool developed to convert xml into Excel format, which enables it to be graphed. In order to automate the graphing process, Excel macros were developed that generate graphs from the created Excel files. Following some interpretation these graphs quantitatively detailed several areas of programmer behaviour displayed during the study.

## 2. Components

The monitoring software contains three separate components, Watcher, Xml2Excel and WatcherMacros, each of which will be briefly discussed in relation to their usage as part of the example pilot study. The combination of these components yields a detailed dataset that can be used to pinpoint areas of dissimilarity between different programmers and programmer techniques. Information is stored with a high-level of detail so decisions as to what information regarding different patterns of programmer behaviour should be considered may be postponed until later post-mortem analysis.

	16	11	16	21	26	31
DoodlePad.java	0.040367		0.120317		1.1026	0.132817
DrawPad.java	0.081517	0.527083	0.060933	1.059633	1.17085	0.014067
OptionPanel.java	0.707017		0.346617	1.085667	1.2758	0.24895
ColorChanger.java	1.13305		0.038283	3.116117	3.9883	0.124217
JColorChooser.class	1.920833					

N.B. Each row represents a line number and each column represents five lines of code. The numerical value represents the amount of time spent throughout the session in that section of code

**Figure 2.0**

## 2.1 Watcher

Watcher consists of an Eclipse plug-in, which generates an xml file for every Eclipse session. The plug-in's execution begins once the Eclipse environment is launched, and runs seamlessly in the background. To maximise ecological validity there is no user interface and the programmer is not required to interact with the plug-in any manner. Exclusion of a user interface ensures the user remains in as natural an IDE environment as possible.

Once Eclipse is launched, the plug-in creates a new session xml file, which stores all the captured events. Each active editor that is a

1. Once Eclipse is launched, a handle on the workbench is obtained.
2. A window listener is then added to the workbench.
3. When a part is activated, it is checked to see if it is an editor.
4. If it is an editor, the source viewer is acquired from it.
5. The widget can then be got from the source viewer.
6. A KeyListener and MouseListener are then added to the widget.
7. The Scrollbar is attained and a SelectionListener is added to it.
8. The Document is then attained and a DocumentListener is added to it.

	Browsing	Scrollbar	MouseEvent	Coding
DoodlePad.java	0.26095	0.535667	0.801817	0.526567
DrawPad.java	0.186483	1.378917	1.189333	1.776783
OptionPanel.java	0.368233	0.15105	0.944033	5.1213
ColorChanger.java	0.472917	0.085933	3.789583	4.839317
JColorChooser.class	0.220317	0.8461	0.615383	0

N.B. Each column represents a file worked on during the session. Each column represents an event. The numerical values represent the amount of time in minutes undertaking a particular event in a file throughout the session.

**Figure 2.1**

part of the session stores each event with the associated line number and time stamp. For example a "keystrokes" event is stored in the following format.

```
<File name="MainClass.java">
  <Event type="KeyEvent" action="page down"
lineNumber="30" time="1077129558515"/>
</File>
```

The manner in which the events are captured and stored is as follows.

At the time of development Eclipse 2.1 was used. In Eclipse 2.1 getSourceViewer() method is protected so reflection is used to get an instance of this method.

## 2.2 Xml2Excel

The Watcher tool captures all information possible in a real-time setting. The XML2Excel tool consists of a post mortem analysis tools that abstracts the particular data required for that study. In this study we were interested in observing the different behavioural patterns taken by the

programmers in question. We were also interested in seeing where the developers spent most of their time during the development process.

In an effort to convert the data to our required format, the xml was processed and converted to excel form where it was graphed. Two types of graphs were chosen to represent the data, namely general graph and events graph. The general graph gives an overview of the time spent in a section of code throughout each file that is worked on. With regards to the general graph, time spent on each line of code is required to graph appropriately. This is calculated by subtracting the time of the previous event from that of the current event. On graphing this data, one problem was evident. Periods of inactivity produced unsightly spikes, which lead to imprecise and distorted data about developer activity. Thus, omission of periods of developer inactivity is essential in order to portray a more accurate

account of time the developer spent working within the environment. This is accomplished by disregarding an event if another event doesn't occur for five minutes or more, the event disregarded is seen as insignificant. The events graph presents the time spent coding, browsing, using the mouse and scrollbar in each file. Time spent coding is calculated by any event that changes the document, browsing is using the arrow keys or page up and page down. At a quick glance the events graph can portray what the developer was doing and where they were doing it. Figure 2.0 and 2.1 shows examples of each type of table.

## 2.3 WatcherMacros

Figures 2.0 and 2.1 can be graphed using a third party graphing tool. Macros were written to automate this process. The graphs produced from the above tables can be seen in Figure 2.3 and 2.4.

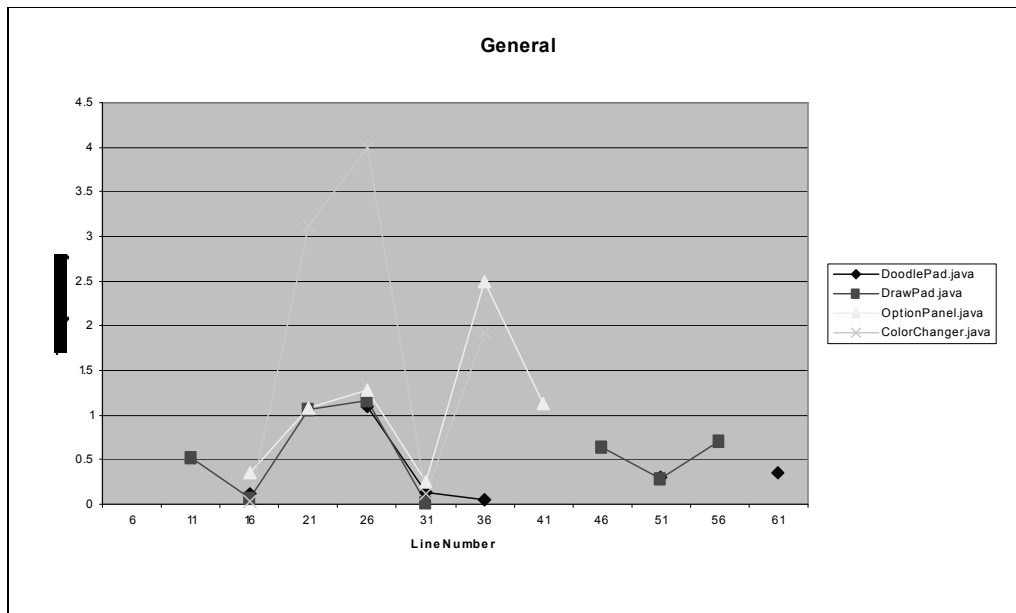


Figure 2.3 Example of General Graph\*

\*This is only a section of the code graphed. The macro allowed the user to choose to graph a section or all the code. From the above graph, it is clear from that the programmer had difficulty in ColorChanger.java as this is where they spent their time.

### 3. Utilisation

As now filtering of data is applied during the capture process the tool can accommodate a number of uses. It can be exploited from both an educational and industrial perspective. Each of these will be described in the following sections.

in the experiment. From figure 2.1 it can be seen that the most time was spent coding in ColorChanger.java. By using the general graph it can be seen that most of the time spent coding was between lines 16-31.

One thing that is noticed at the event graph (Figure 2.4) is the presence of the JColorChooser.class. This is obviously a

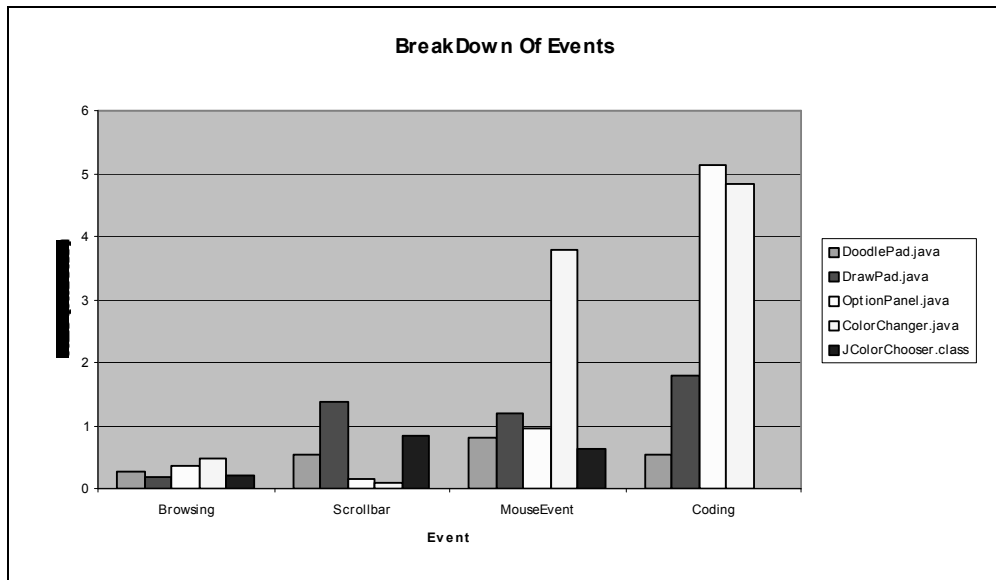


Figure 2.4 Example of Events Graph.  
\*Here we can see that most of the time coding was in ColorChanger.java.

feature in eclipse that participant A was aware of whereas the other participants either didn't use or weren't aware of it. That is of course the ability to open up existing java source in the Java API to see how it is implemented. Also noticed was that Participant A spent significantly less time browsing the code than Participant C. This might be attributed to the experienced developer's comprehension time being much shorter than that of the less experienced developers.

### 3.1 Educational

For educational studies it can be used as an aid in pinpointing areas where students are having difficulties and spending most of their time. These areas can then be addressed in tutorials and lectures.

An experiment was run with three separate developers of varying levels of ability. These included a fourth year Computer Science student Participant A and a second year Computer Science student Participant C. The findings of the experiment are discussed below.

The graphs above and tables above are from the most experienced developer who took part

### 3.2 Industry

In industry it can be used as a way of highlighting areas where developers spent the most time, these areas could then be studied to identify possible areas of complexity or unstructured code that may need refactoring. It may be useful in the analysis of a project in relation to bugs that occurred in the project. It would be interesting to see if there is a correlation between the bug count of a specific

section of code and how this relates to the time spent in that section. To the best of our knowledge this is an aspect of maintenance that as of yet has not been studied.

## 4. Conclusion

The tool could be exploited in many areas.. One of which has been described above. Aside from its possible use as a teaching aid for an undergraduate-programming course. It may provide an excellent platform in the monitoring of other Eclipse plug-in usage either in isolation or in combination. It may be interesting to observe how programmer's behaviour changes in relation to the introduction of a new capability provided by a future innovative plug-ins.

## 5. References

[1] Brooks, R. E. (1980) Studying Programmer Behaviour Experimentally: The problems of Proper Methodology. *Communications of the ACM*, 23, 207 - 213.

[2] Eclipse Platform Technical Overview Object Technology International, Inc. February 2003 (updated for 2.1; originally published July 2001)

[3] Romero, P., Cox, R., du Boulay, B., and Lutz, R. (2002) Visual attention and representation switching during java program debugging: A study using the restricted focus viewer. In *Diagrams 2002: Second International Conference on Theory and Application of Diagrams*, number 2317 in *Lecture Notes in Artificial Intelligence*, pages 221-235. Springer-Verlag.

[4] Sheil, B. (1981) The Psychological Study of Programming. *Computing Surveys*, 13, 101 - 120

## 5.1 Author Biographies

John McKeogh, Honours Degree in Computer Systems, University of Limerick. Currently employed by Symantec, involved in QA automation and localization. John would like to thank everyone who helped him throughout his studies especially those who participated in the pilot study, David Dineen, Richard Fanning and Kieran O'Mahony.

Chris Exton is currently a lecturer in the Department of Computer Science and Information Systems at University of Limerick. His research interests include software tools and programmer comprehension.