Mining a Developer's Workflow from IDE Usage

Constantina Ioannou

Kongens Lyngby 2017
M.Sc.-2017-

Abstract

Software developers interact with integrated development environments (IDEs) by issuing commands that execute various programming tools, from source code formatters to build tools. Several studies has shown that despite the usefulness of provided tools and capabilities to perform tasks such as: navigating among classes and methods, continuous compilation, code refactoring and integrated debugging, IDEs usually tend to overload developers resulting a chaotic state. In other words, commonly developers spend time shifting between several artifacts of the working environment in order to reach their goal, and this repeated process increases the complexity of identifying the relevant information to solve their assignment.

For this reason, an assortment of usage metrics plugins to gather developers' activity have been developed in an attempt to understand a developers' workflow. Understanding the workflow, as well as the skillset and experience of a developer is the first step into improving their productivity and reducing the unneccessary complexity created within IDEs.

In this work, we analyze and provide an overview of available existing plugins for recording a developer's interaction within an IDE and we further enhance a tool in order to capture, mine and analyze the process of software developement. We aim to contribute information for modelling an automatic activity detection tool capable of tracking the workflow of developers' activity which potentially will be used as input in later studies for a reconfiguration program supporting the developer.

In more depth, this thesis will provide an overview of contextual factors and an analysis of how they can be achieved using already existing plugins that allow

recording developers' activity within Eclipse IDE. In addition, is expected to uncover and characterise developer's workflow. *TODO: write*

Preface

This thesis was carried out at the department of DTU Compute, at the Technical University of Denmark, in fulfilment of the requirements for acquiring an M.Sc. in Computer Science and Engineering. The goal of this project was to collect data on how developers interact with the IDE while developing software and to mine developers' workflows from IDE usage using process mining techniques. This thesis project was an extention of a previous plugin of eclipse called Rabbit, which is a metric tool for user interaction with the IDE of Eclipse.

This report describes the project itself, discusses the concepts involved, describes in detail the software that was used and developed during this project, and presents the results of mining developers' interactions with the IDE of Eclipse. *TODO: add signature*

Acknowledgements

I would like to thank...

Contents

Chapter 1

Introduction

The process of software development doesn't feel any better than it did a generation ago. Yet researchers pointed out certain aspects of the problem still the workflow and the way a progrmamer might thing at a particular moments is still a mystery a black box

understanding how they do it, and how they keep the dtructures, the strategies and how the tools are used is not an easy task. in order to improve the environments available we have although to understand these aspects.

ides aim to help the user however often they create a general chaos with all the navigation the studies of .. and the Cognition studies and models strategies top down bottom up,

previous research of understanding the workflow of a developer is done by ...

previous research of attempting to extracct user activities from Integrated Development Environment (IDE) ...

1.1 Integrated Development Environment

1.2 The problem

1.3 The approach

1.4 Structure of thesis

CHAPTER 2

The Eclipse IDE

In the following chapter a general overview of Eclipse framework and its most prominent components is provided. A description and analysis of available plug-ins from which essential information could be captured is presented. Last but not least a great amount of focus is given on a specific plug-in called Rabbit, which is used and enhanced throughout the thesis.

2.1 Eclipse

Eclipse workbench is a powerful UI framework for IDEs fairly used by several developers: to primarily develop Java applications or even to develop in other programming languages, and to also develop documents and packages for other softwares. It provides several services for highly integrated and extensible user interfaces.

The standard SDK Eclipse distribution contains a base workspace with certain functionalities, Java Developemnt tooling plug-ins and layout. Thereafter additional plug-ins can be included or created to allow the extension of the workspace. As a result Eclipse can become a multifunction framework to be used for: Embedded programming, C++ programming, javaBeans, Java application, websites, or even develop additional Eclipse plug-ins, etc. The plug-ins can be removed or replaced according to the needs and preferences of developers.

Even though Eclipse framework started as a replacement for Visual Age for Java from IBM ADD Reference: give reference wikipedia, it became an open platform for integrating tools, editors, views and plug-ins. Its source code is freely available and anyone can contribute by building their own new plug-ins or by engaging in discussions regarding integrated tools.

The most significant reasons of selecting the Eclipse IDE as a tool for gathering information for processing developers' workflow are:

- A huge portion of java developers use Eclipse IDE, as mentioned in ADD Reference: REFERENCE BOOK
- Most of the features and plug-ins implementd for Eclipse are available online, and Eclipse itself is an open source framework

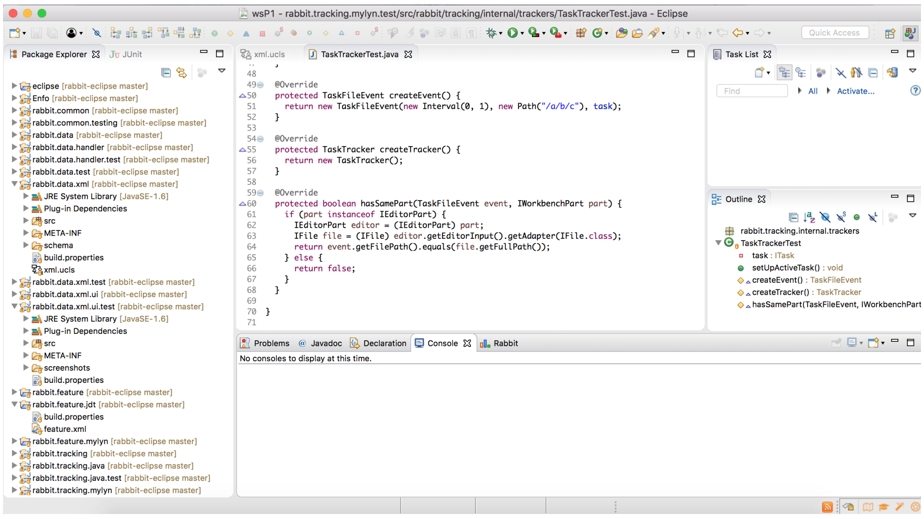
2.2 Eclipse User Interface

2.2.1 Eclipse UI as Application Developer

ADD Reference: The eclipse foundation. eclipse documentation:release oxygen. online url// eclipse.org Eclipse framework consists of workbench windows, which have integrated basic elements such as active perspectives, editors, views and menus.

When starting up Eclipse, if its a newly initiated greets the developer with a welcoming screen, otherwise the previously workspace is loaded. Figure **??** displays a standard Eclipse setup for a Java IDE.

A *perspective* has its own a set of elements views and editors and menus along with a adaptable personalized layout for specific tasks such as debugging a program. A *view* is a window that enables the developer to examine something, eclipse offers various of views f.x. Java packages and their files. An *editor* is a smart editor which recognizes the programming language markups the syntax and allows you to modify and save files. Editors share characteristics with views, but unlike views, editors don't have tool-bars. Eclipse is filled up with *menus*, the main menu is at the top of the screen while views and explorers usually provide their own context menus.



```
eclipse [rabbit-eclipse master]
Enfo [rabbit-eclipse master]
rabbit.common [rabbit-eclipse master]
rabbit.common.testing [rabbit-eclipse master]
rabbit.data [rabbit-eclipse master]
rabbit.data.handler [rabbit-eclipse master]
rabbit.data.handler.test [rabbit-eclipse master]
rabbit.data.test [rabbit-eclipse master]
rabbit.data.xml [rabbit-eclipse master]
  JRE System Library [JavaSE-1.6]
  Plug-in Dependencies
  src
  META-INF
  schema
  build.properties
  xml.ucls
rabbit.data.xml.test [rabbit-eclipse master]
rabbit.data.xml.ui [rabbit-eclipse master]
rabbit.data.xml.ui.test [rabbit-eclipse master]
  JRE System Library [JavaSE-1.6]
  Plug-in Dependencies
  src
  META-INF
  screenshots
  build.properties
rabbit.feature [rabbit-eclipse master]
rabbit.feature.jdt [rabbit-eclipse master]
  build.properties
  feature.xml
rabbit.feature.mylyn [rabbit-eclipse master]
rabbit.tracking [rabbit-eclipse master]
rabbit.tracking.java [rabbit-eclipse master]
rabbit.tracking.java.test [rabbit-eclipse master]
rabbit.tracking.mylyn [rabbit-eclipse master]
```

```java

@Override
protected TaskFileEvent createEvent() {
  return new TaskFileEvent(new Interval(0, 1), new Path("/a/b/c"), task);
}

@Override
protected TaskTracker createTracker() {
  return new TaskTracker();
}

@Override
protected boolean hasSamePart(TaskFileEvent event, IWorkbenchPart part) {
  if (part instanceof IEditorPart) {
    IEditorPart editor = (IEditorPart) part;
    IFile file = (IFile) editor.getEditorInput().getAdapter(IFile.class);
    return event.getFilePath().equals(file.getFullPath());
  } else {
    return false;
  }
}

}

```

```
rabbit.tracking.internal.trackers
TaskTrackerTest
  task : ITask
  setUpActiveTask() : void
  createEvent() : TaskFileEvent
  createTracker() : TaskTracker
  hasSamePart(TaskFileEvent, IWorkbenchPart
```

```
No consoles to display at this time.
```

Figure 2.1: The Java Development perspective in Eclipse.

Eclipse IDE is highly adjustable and enables the developer to construct and save his own perspectives according to his preferences. Having these advantages in mind one can optimise working capabilities.

2.2.2 A closer look to Eclipse

A closer look into Eclipse and how it works is provided. Each workbench is consisted by several windows as mentioned. These windows have their own selection service instance to be handled. The selection service instances are responsible to track selection activity and to propagate selection changes to all registered listeners. Selection events can be triggered either through user interaction or programmatically, and they occur when the selection in the current active part is changed or when a different part is activated.

The view where elements or text is selected does not need to know who is interested in the selection. This gives the capability of creating new views that depend on the selection of an existing view without changing the context of the existing view.

The next sections explain who provides what kind of selections to whom.

2.3 Plug-ins

2.3.1 Plug-in Structure

Eclipse is a multifunction framework and it provides several software components (plug-ins). Eclipse Marketplace client gives the advantage of searching, and discovering popular extensions for developers to enhance their working environment. Eclipse is based on the OSGi technology, and the software components are usually packaged and distributed as OSGi bundles. OSGi bundles are very similar to standard JAR packages. An OSGi bundle must contain a manifest file with the mandatory metadata.ADD Reference: Richard hall et al osgi in action creating modular applications in java 1st greenwich ct usa manning publication co 2011 isbn 1933988916, 9781933988917. This metadata includes a name, version, activator, dependencies, API, etc.

The Plug-in Developer Environment (PDE) enables developers to extend their Eclipse IDE by creating new additional functionalities or even improve the capabilities of already implemented plug-ins. Although the reason Eclipse provides this thumping range of tools is to support developers, usually instead a chaotic environment is generated. For this reason several researchers and developers instrumented plug-ins with the aim of gathering information for developers' activity, and also to further develop Recommendation System in Software Engineers (RSSE)s.

2.3.2 Plug-in Analysis

For this thesis the plug-ins in concern are mostly observing the usage of Eclipse artifacts and recording activity as well as interactions of a developer. Several of these type of plug-ins can be found in git-hub and the Eclipse Market, emphasize was given to the following plug-ins:

Fluorite is a plug-in developed in the School of Computer Science at Carnegie Mellon university. Its purpose is low level event logging for Eclipse when using the code editor. In other words, events such as: ADD Reference: add link http://www.cs.cmu.edu/ fluorite/ character type, text cursor movement, selected text modification, as well as, all the other available Eclipse commands that can be called for an editor. Fluorite could potentially be used to extract developers' coding activity and use to detect and measure the time for various usage patterns or events of interest. However, this plug-in seemed promising and a good candidate for process mining, it was not possible to gain access to its source code.

Usage Data collector is a framework for collecting information about how developers are using the eclipse platform. Information for views usage, editor usage, changes of perspectives, actions invoked are recorder by monitors which are being used including also a timestamp. Moreover this plug-in also collects basic information about the runtime environment (OS, system architecture, window system, locale, etc). This information are uploaded periodically to servers hosted by The Eclipse Foundation with the purpose to be processed in a later stage ADD Reference: https://www.eclipse.org/org/usagedata/. Since this plug-in records relevant information it could offer a useful structure for mining the developers workflow, however, due to the fact that the development stopped a long time ago, access to source code was not achieved. The project was shut down mainly due to resource constraints. Despite the efforts of researchers, not valuable knowledge was obtained from the data.

Metrics is a plug-in that calculates various metrics for your code during build cycles and warns you of range violations for each metric. ADD Reference: https://marketplace.eclipse.org/content/eclipse-metrics. Metrics examples are the number of classes, children, interfaces, depth of inheritance, overridden methods,etc. This information can be extracted as an xml and therefore some data mining could be performed.

Mylyn is a subsystem in Eclipse used for task management. The original name of this project is Mylar, and was used in studies such as ADD Reference: how are java softwaere developers usingthe eclipse IDE because it captures events, such as preference changes, perspective changes, window events, selections, periods of inactivity, commands invoked through menus or key bindings and URLs viewed through the embedded Eclipse browser. This tool allows the developers to work in a task-focused interface, such tasks are f.x. fixing bugs, problem reports, new features. Mylyn allows developers' tasks to be organized and monitors its activity. It provides awareness for the progress of tasks, and increases the productivity by reducing unnecessary navigation, searching and scrolling. Further Mylyn allows the creation of graph elements and relationships of program artifacts.
TimeKeeper this plug-in was an extension to Mylyn in order to track time, and report how long a developer worked on a certain task.

Rabbit is a statistics tracking plug in for Eclipse. Rabbit collects data on which operation have been performed over a set period of time. Rabbit view provides developers with information regarding their command usage (copy,paste, build) used as well as their tools usage (resources, perspectives, launches). The development of Rabbit was also terminated, however documentation and source code were available on github. Rabbit was selected to further explored, more detailed analysis is provided in 2.4.

ITrace interfaces with an eye tracker, determines the location of eye gaze and map to source code element

2.4 The focal plug-in: Rabbit

Bibliography