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' work-flow. 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 IDEs ...

In this chapter we introduce some key aspects of Real-Time Systems (RTS), starting from the computational model. We then discuss *timing analysis* and Worst-Case Execution Time (WCET) calculation. After that we introduce *time-composability* and how important that property is. In Section **??** the role of a Real-Time Operating System (RTOS) is described and some examples are presented. In the end the T-CREST project and platform are briefly described.

1.1 Integrated Development Environment

TODO: list available and discuss TODO: paragraph why we choose eclipse TODO: programmer workflow within IDE

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 plugins from which essential information could be captured is presented. Last but not least a great amount of focus is given on a specific plugin which is used and enhanced throughout the thesis.

TODO: ide eclipse plugins, available what are useful, present all, write adv and dis. describe contextual factors that can be achieved, describe how programmers work within this ide. TODO: focus on rabbit specia chapter explain why to choose that, positive negative, limitations. provide class diagram and explanation, name the issues

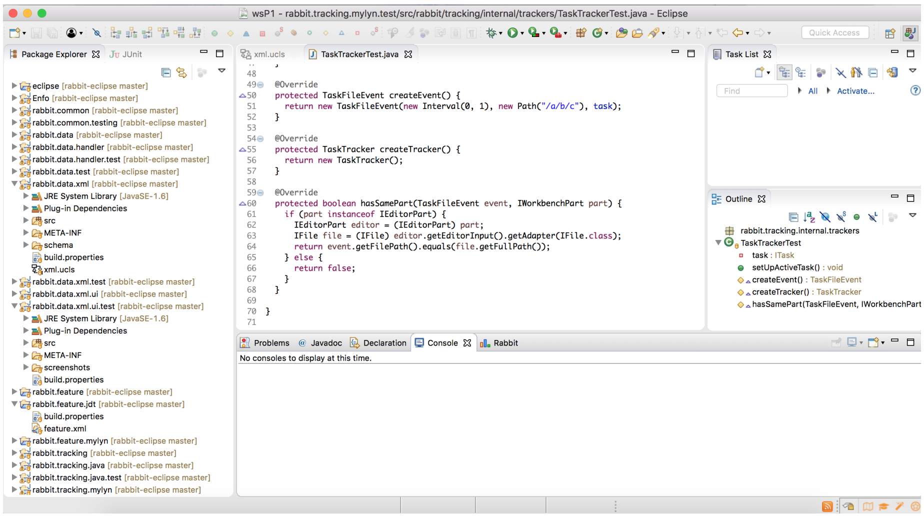
2.1 Eclipse

Eclipse framework is an IDE fairly used by several developers: to primarily develop Java applications, to also develop in other programming languages via plugins, and to develop documents and packages for other softwares. The standard SDK Eclipse distribution contains a base workspace with certain functionalities, Java Developemnt tooling plug-ins and layout. Thereafter additional plug-ins allow the extension of the workspace, and Eclipse can become a multifunction framework to be used for: Embedded programming, C++ programming, javaBeans, Java application, websites, or even develop additional Eclipse plugins, etc. The plug-ins can be removed or replace according to the needs of the developer.

Eclipse framework is an open platform for integrating tools, editors, views and plug-ins. Its source code is freely available and anyone can contribute by builting their own new plug-ins or by engaging in discussions regarding integrated tools.

2.2 Menus, views and editors

When starting up Eclipse, either a welcoming screen greets the developer if its a newly initiated or the previously workspace is loaded otherwise. Figure **??** displays a standard Eclipse setup, a java integrated development environment. Eclipse's main window is called the *workbench* and based on the active *perspective* its appearance is defined. A perspective has its own a set of elements views and editors and menus along with a adaptable personalised layout for specific tasks such as debugging a program. A view is a window that enables the de-


wsP1 - rabbit.tracking.mylyn.test/src/rabbit/tracking/internal/trackers/TaskTrackerTest.java - Eclipse
Quick Access
Package Explorer
JUnit
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]
xml.ucls
TaskTrackerTest.java
Task List
Find
All
Activate...
Outline
rabbit.tracking.internal.trackers
TaskTrackerTest
task : ITask
setUpActiveTask() : void
createEvent() : TaskFileEvent
createTracker() : TaskTracker
hasSamePart(TaskFileEvent, IWorkbenchPart
Problems
Javadoc
Declaration
Console
Rabbit

```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;
      }
    }

  }

```

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

Figure 2.1: The Java Development perspective in Eclipse.

veloper to examine something, eclipse offers various of views f.x. Java packages and their files. An editor is a smart editor which recognises 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 toolbars. Eclipse is filled up with menus and toolbars, the main menu is at the top of the screen while views and explorers usually provide their own context menus.

2.3 Plug-ins

Eclipse is a multifunction framework and it provides several software components (plug-ins). By using the Plug-in Developer Environment (PDE) developers can either extend Eclipse IDE by creating new additional functionalities for plug-ins or even improve and extend the capabilities of already implemented plug-ins. When a plugin is installed the platform dynamically discovers the registered plug-ins and invokes them when they are needed.

Adding own plug-ins starts with deciding how the plug-in will be integrated with the Eclipse platform, e.g. the main menu or toolbar or a context menu. The extensions associated with these user interface items have to be implemented, and a declaration of the implementation's classes and extension points will have to be provided in an XML-format plug-in manifest file (called plugin.xml). More information on developing plug-ins can be found on the Eclipse main website1 and in the Eclipse platform's help files

2.4 the focus for rabbit

Bibliography