SOFTWARE SPLab PRACTICES

Code History Questionnaire

Consent Form

Welcome to our survey on source code history. This first page provides some details and background information and asks for your consent to participate.

Who is conducting the study?
Principal Investigator: Dr. Reid Holmes, Associate Professor, Department of Computer Science, UBC, rtholmes@cs.ubc.ca, 604-822-0409.
Co-Investigator: Felix Grund, Graduate Student, Department of Computer Science, UBC, ataraxie@cs.ubc.ca.

Who is funding the study?
This study is funded by NSERC.

Why are we doing this study?
You are being invited to participate in a survey that investigates how source code history can be improved by support for dedicated history of semantic code units. The results of the survey will inform the development and evaluation of a prototype version control tool.

How is the study done?
Your participation in the study will involve answering a questionnaire regarding your experience with source code history tools, both with general questions and real-world scenarios. The study will take approximately 20 minutes to complete.

What happens next?
The aggregate results of the study will be made available through open channels and may be published in peer reviewed journals without any individual respondent or institutional identifiers.

Is there any way the study could pose a risk for you?
There are no anticipated risks for participants. You do not have to answer any questions you feel uncomfortable answering and there is a *don't know* option for all multiple choice answers. You can end the study at any time with

no repercussions. The study will only ask very limited questions about your professional background as a software developer and your experience with source code history tools.

What are the benefits of participating in the study?
The survey will provide you with an opportunity to provide valuable feedback for the development of a novel version control tool. Additionally, you can enter a lottery for one $100 (CAD) Amazon gift card by providing your email address at the end of the survey.

How will your privacy be maintained?
Your confidentiality will be respected. Original data collected in this study will be examined by the research team members only. Although limited identifying information will be collected, the research team will ensure that any instances of self-disclosure will be anonymized. Reports will contain descriptive statistics and include select quotes with any identifiers removed. The reports will not contain any personally identifying data. The research team will not identify individuals in publications. The official UBC survey tool we are using complies with the BC Freedom of Information and Protection of Privacy Act (FIPPA) because the survey data is kept secure and is stored and backed up in Canada.

Who can you contact if you have questions about this study?
If you have any questions or concerns about what we are asking of you, please contact the co-investigator. Contact information is listed at the top of the first page of this form.

Who can you contact if you have complaints or concerns about this study?
If you have any concerns or complaints about your rights as a research participant and/or your experiences while participating in this study, contact the Research Participant Complaint Line in the UBC Office of Research Ethics at 604-822-8598 or if long distance e-mail RSIL@ors.ubc.ca or call toll free 1-877-822-8598.

Taking part in this study is entirely voluntary. You have the right to refuse to participate in this study. If you decide to take part, you may choose to pull out of the study at any time without giving a reason and without any negative impact on you or your employment.

By clicking “Continue”, you confirm to have read the survey consent form and consent to participate.

Introduction

Introduction

Software systems evolve over time. This evolution can often be traced through version control systems. These systems provide mechanisms to browse through a file’s changes, but do not provide support for navigating semantic code units like classes, methods, and fields. Additionally, these mechanisms are not effective in the face of common code modification tasks like refactoring.

The purpose of this survey is to determine whether and how developers use source code history and how history navigation tools can better support the kinds of tasks developers are trying to perform. This survey is intended for software developers with some experience in source code history (e.g. commit diffs, pull requests, file history). It does not require experience in a specific programming language. It will take approximately **20 minutes** and can be paused and resumed any time. Please answer as many questions as possible and select the *don't know* option rather than submitting empty responses.

Section 1: general questions

Section 1: Source Code History

In this section we aim to collect a general understanding how developers use source code history. By *source code history*, we are referring to any activity, system, or tool associated with past changes to source code; common examples are the history of a specific file, commit diffs or pull requests.

Q1.1 How recently did you last use source code history of any kind?

- Less than 2 work days
- Less than 1 week
- Less than 1 month
- Less than 1 year
- More than 1 year
- can't remember

Q1.2 Please describe this most recent activity. How did you use source code history? What were you looking for? Did you find it? Did the tools you used support you in this investigation or could they be improved?

Q1.3 In terms of source code granularity, how interested are you in gathering information on source code history at the following levels?

	Very interested	Interested	Neutral	Not very interested	Not interested at all	Don't know
Project						
Directory/Package						
File						
Class/Module						
Field/Variable						
Method/Function						
Block*						

* By *block* we are refering to a group of declarations or statements that we commonly see between curly braces ({}) or keywords like *begin*/*end* in programming languages.

Q1.4 When you use code history, how far in the past do you usually examine? How do you determine how far in the past you want to go?

Section 2: pull request scenario

Section 2: Pull Request Example

For the following questions, imagine the following situation: you are reviewing a change to a code fragment in a pull request* but you are not certain about what the code is actually doing. Your goal is to better understand the code and what led to the change being made.

**Pull request* is a common term in version control for the activity of a source code contributor requesting that a project maintainer merges a change into the code base of the project. If you are unfamiliar with this terminology, you can simply assume a simple change to the source code base that you are reviewing.

Q2.1 Does this scenario sound familiar to you (i.e. have you encountered this in the past)?

Very familliar	Familiar	Neutral	Not very familiar	Not familiar at all	Don't know
○	○	○	○	○	○

Q2.2 Please describe very briefly how you would approach this problem. What kinds of questions would you like to answer? What tools or approaches would you use to answer them?

Suppose the change you are reviewing is related to a single method. You want to understand this method better and what led to it being changed.

Q2.3 Using source code history, how would you find changes to **this method only**? Please describe briefly.

Q2.4 How well would your strategy cope with more complex structural changes, e.g. method renaming, moving of a method, refactoring?

	Very well	Well	Neutral	Not very well	Not well at all	Don't know
Renaming of method	O	O	O	O	O	O
Signature changes (parameters, return type)	O	O	O	O	O	O
Move to a different file	O	O	O	O	O	O
Splitting into multiple methods	O	O	O	O	O	O
Combinations of the previous	O	O	O	O	O	O

Q2.5 Using current tooling support, how hard is it generally to trace changes to a specific method?

Very hard	Hard	Neutral	Not very hard	Not hard at all	Don't know
O	O	O	O	O	O

Q2.6 Given your answer to the previous question (Q2.5), what makes this hard or easy?

Section 3: Historical Scenario Overview

Section 3: Specific Scenario - Overview

We have chosen a specific scenario that illustrates how source code history relates to development in practice. Please read the description below and answer the questions that follow. Please allow a few minutes and click on the links provided to understand the scenario better. The choice of the Java language for the example is arbitrary and does not require Java experience.

Example Scenario

Imagine yourself in the dev team of Checkstyle, a popular syntax checker for Java. You are to review this pull request with a change to the method *CommonUtils.hasWhitespaceBefore*. In order to review this pull request, you want to get a better picture on this method and how it has changed over the past. You decide to look into the history of the file *CommonUtils.java* as seen here. You discover that this file has a history of 47 revisions in 3 years.

Q3.1 In the above version history, how would you identify the commits in which the method of interest has changed? Please describe your strategy briefly.

Q3.2 How well do existing tools support identifying these changes?

Very well	Well	Neutral	Not very well	Not well at all	Don't know
O	O	O	O	O	O

Q3.3 How useful would it be to have support for a more semantic history in this scenario (e.g. history for this method or class only)?

Very useful	Useful	Neutral	Not very useful	Not useful at all	Don't know
O	O	O	O	O	O

Q3.4 How hard would it be to find the first commit for the given method and whether the method was really created then or if it was moved there from somewhere else (e.g. through a file renaming, or through a refactoring)?

Very hard	Hard	Neutral	Not very hard	Not hard at all	Don't know
O	O	O	O	O	O

Section 4: Historical Scenario Detail

Section 4: Historical Scenario Detail

For the same real world example as above (Checkstyle pull request), we have analyzed the version control history of the method of interest. The following descriptions and diff snippets show where and how the method has been changed over the past. Please have a look at these and answer the questions that follow.

Example Scenario (details)

The history of the file CommonUtils.java shows that a refactoring commit on Aug 28 2015 (46a52f8) renamed the method *whitespaceBefore* to *hasWhitespaceBefore:*

```
93  -    public static boolean whitespaceBefore(int index, String line) {
94           for (int i = 0; i < index; i++) {
95               if (!Character.isWhitespace(line.charAt(i))) {
96                   return false;
```

```
93  +    public static boolean hasWhitespaceBefore(int index, String line) {
94           for (int i = 0; i < index; i++) {
95               if (!Character.isWhitespace(line.charAt(i))) {
96                   return false;
```

This is the third-oldest commit in the file's history. The message of the oldest commit in the file's history on Aug 26 2015 (cdf3e56) is "Utils class has been splitted to CommonUtils and TokenUtils". The diff confirms that a file Util.java was split into these two separate files CommonUtils.java and TokenUtils.java and that the method *whitespaceBefore* came from this file:

```
41  /**
42   * Contains utility methods.
43   *
44   * @author [redacted]
45   */
46  -public final class Utils {
```

```
34  /**
35   * Contains utility methods.
36   *
37   * @author [redacted]
38   */
39  +public final class CommonUtils {
```

```
143       /**
144  -     * Returns whether the specified string contains only whitespace up to the
145  -     * specified index.
146       *
147  -     * @param index index to check up to
148  -     * @param line the line to check
149       * @return whether there is only whitespace
150       */
151      public static boolean whitespaceBefore(int index, String line) {
@@ -158,10 +100,12 @@ public static boolean whitespaceBefore(int index, String line) {
158       }
```

```
84        /**
85   +     * Returns whether the specified string contains only whitespace up to the specified index.
86        *
87   +     * @param index
88   +     *            index to check up to
89   +     * @param line
90   +     *            the line to check
91        * @return whether there is only whitespace
92        */
93       public static boolean whitespaceBefore(int index, String line) {
100       }
```

Inspecting the history of Utils.java reveals 41 revisions throughout the year 2015. However, in the oldest commit from Jan 21 2015 (204c073), the method *whitespaceBefore* was not present in this file. Searching for the commit that introduced the method reveals a commit from March 15 2015 (1c15b6a) with the message “move all methods from checkstyle.api.Utils to checkstyle.Utils”. Again, this was a refactoring commit that combined two classes with the same name (Utils.java) to one file:

```
40  -/**
41  - * Contains utility methods.
42  - *
43  - * @author [redacted]
44  - */
45  -public final class Utils
46  -{
```

```
72  -    /**
73  -     * Returns whether the specified string contains only whitespace up to the
74  -     * specified index.
75  -     *
76  -     * @param index index to check up to
77  -     * @param line the line to check
78  -     * @return whether there is only whitespace
79  -     */
80  -    public static boolean whitespaceBefore(int index, String line)
81  -    {
82  -        for (int i = 0; i < index; i++) {
83  -            if (!Character.isWhitespace(line.charAt(i))) {
84  -                return false;
85  -            }
86  -        }
87  -        return true;
88  -    }
```

```
40  /**
41   * Contains utility methods.

45  public final class Utils
46  {
```

```
108 +    /**
109 +     * Returns whether the specified string contains only whitespace up to the
110 +     * specified index.
111 +     *
112 +     * @param index index to check up to
113 +     * @param line the line to check
114 +     * @return whether there is only whitespace
115 +     */
116 +    public static boolean whitespaceBefore(int index, String line)
117 +    {
118 +        for (int i = 0; i < index; i++) {
119 +            if (!Character.isWhitespace(line.charAt(i))) {
120 +                return false;
121 +            }
122 +        }
123 +        return true;
124 +    }
```

The history of the old Utils.java file from which the method came reveals 69 revisions with the first one dating back to Feb 20 2002 (e10faf3). The details of this commit show that the method *whitespaceBefore* was introduced in this commit for the first time.

```
+final class Utils
+{
+    /** stop instances being created **/
+    private Utils()
+    {
+    }
+
+    /**
+     * Returns whether the specified string contains only whitespace up to the
+     * specified index.
+     *
+     * @param aIndex index to check up to
+     * @param aLine the line to check
+     * @return whether there is only whitespace
+     */
+    static boolean whitespaceBefore(int aIndex, String aLine)
+    {
+        for (int i = 0; i < aIndex; i++) {
+            if (!Character.isWhitespace(aLine.charAt(i))) {
+                return false;
+            }
+        }
+        return true;
+    }
```

Q4.1 Consider again the described situation of being faced with a pull request for a change of a method. How helpful would you consider the information above for getting a better understanding of the method and its history?

Very helpful	Helpful	Neutral	Not very helpful	Not helpful at all	Don't know
O	O	O	O	O	O

Q4.2 How hard would you consider retrieving information on the history of a method with the above level of detail?

Very hard	Hard	Neutral	Not very hard	Not hard at all	Don't know
O	O	O	O	O	O

Q4.3 If a tool could generate information in the fashion of the above on any method or other code unit, how valuable would you consider this tool?

Very valuable	Valuable	Neutral	Not very valuable	Not valuable at all	Don't know
O	O	O	O	O	O

Q4.4 What other information that is not in the descriptions above would you consider valuable?

Section 5: Background Information

Background Information

Q5.1 How many years have you been programming?

< 1 year	1-3 years	4-10 years	> 10 years
○	○	○	○

Q5.2 How long have you been working as a professional software developer?

< 1 year	1-3 years	4-10 years	> 10 years
○	○	○	○

Q5.3 How many years have you been using source code version control?

< 1 year	1-3 years	4-10 years	> 10 years
○	○	○	○

Q5.4 What is your current job title?

Q5.5 What version control systems and tools do you use? Please select one or more options.

- ☐ Git
- ☐ Mercurial
- ☐ CVS
- ☐ SVN
- ☐ Github
- ☐ Bitbucket
- ☐ SourceTree
- ☐ IDE/Editor
- ☐ SmartGit/SmartSVN
- ☐ TortoiseGit/TortoiseSVN
- ☐ GitKraken
- ☐ TFS (Team Foundation Server)
- ☐ Visual Studio Online
- ☐ Other (see next question)

Q5.6 If you selected IDE/Editor in the previous question, please specify what IDE/Editor (and if you know, what underlying version control system) you are using. If you selected "Other", please specify what other tools you use.

Q5.7 Do you have any final comments? Do you have any other ideas for tool support or systems to solve the general problems described in this survey and its scenarios? Is there anything else on your mind?

Q5.8 If you are interested in the results of this survey and/or you want to enrol for the $100 (CAD) Amazon gift card lottery, please provide your email address. (Your email address will not be stored with the survey data.)