FeedBaG: An Interaction Tracker for Visual Studio

Sven Amann, Sebastian Proksch, and Sarah Nadi
Software Technology Group
Technische Universität Darmstadt
Email: {amann,proksch,nadi}@cs.tu-darmstadt.de

***Abstract*—Integrated Development Environments (IDEs) provide a convenient standalone solution that supports developers during various phases of software development. In order to provide better support for developers within such IDEs, we need to understand how developers use them. To infer useful conclusions, such information should be gathered for different types of IDEs, for different programming languages, and in different development settings.**

In this paper, we present FEEDBAG, an extension for Visual Studio that tracks developers' interactions with the IDE. FEEDBAG generates a rich stream of interaction events and provides means for developers to review and submit the data to a server. We recently used the tool in a study, recording more than 6,300 hours of work time. Future studies with different user groups are needed to explore and compare IDE-usage aspects, like code-comprehension assistance, in detail. Therefore, we publish FEEDBAG and encourage other researchers to use it as well.

I. INTRODUCTION

Integrated Development Environments (IDEs) are very popular among software developers since they provide support for many of their daily development or maintenance tasks. Modern IDEs provide tools such as specialized code search, task-sensitive views, or quick-peek previews to help developers navigate and comprehend source code. In order to advance this assistance, we need empirical data on how developers use it and how different tools impact this use.

Previous studies investigated developers' use of IDEs [1]–[4]. These studies looked at Java development in Eclipse and Smalltalk development in the Pharo IDE. Recently, we extended this space of knowledge by a large-scale study on the usage of C# development in Visual Studio [5]. We analyzed over 6,300 hours of work time to investigate typical activities and tool usage. We observed that navigation likely indicates a need for code comprehension and identified opportunities by comparing usage patterns between IDEs.

To track how developers use Visual Studio, we developed FEEDBAG, an extension that instruments the IDE and records developers' interactions with it. FEEDBAG is publicly available for any developer to use and is open source. Links to the tool and related material are available on our artifact page [6].

We proceed as follows: Section II describes how FEEDBAG instruments Visual Studio, how it addresses privacy concerns, and how developers submit their data. Section III discusses applications of the tool and Section IV presents related tools.

II. FEEDBAG

To collect information about interactions of developers with Visual Studio, we instrument the IDE. We do this through *ReSharper* (R#) [7], a widely used Visual Studio extension. R# is designed as an extensible platform. It provides access to a semantic model of the source code under edit, including, for example, a type-resolved abstract syntax tree. We implemented FEEDBAG as a R# plugin that generates interaction events and allows users to manage them.

FEEDBAG consist of three parts: The *Event Generator* instruments the IDE and records developers' interactions with it. The *Event Manager* allows developers to review and manage the recorded interactions and to upload them to our server. The *Server* receives, checks, and stores developers' submissions. We subsequently discuss these three components.

A. Event Generator

The Event Generator hooks into every User Interface (UI) control, executable command, window, and editor in Visual Studio and into some system-level events. Whenever the developer interacts with one of these elements, the generator creates an event and stores it on her machine. This process does not interfere with IDE functionality. Each event captures:

Timing When the event started and, if it takes some time to finish (e.g., when a project is built), its duration.

Trigger Whether the event is triggered by a mouse click, a shortcut, while typing, or automatically.

Location The active window and –if available– the active document at the point of the event's generation.

Type Specific information depending on the interaction:

Activity: When the user moves the mouse or scrolls, we record this activity.

Build: When the user runs a build, we capture the target projects, the build duration, and whether it succeeds.

Code Completion: When the user activates the code completion, we capture the proposals, the invocation context, and detailed interactions, like the selection.

Commands: When the user invokes a command, be it native (e.g., save file) or offered by an extension (e.g., R#'s quick fixes), we capture that command's unique Id.

Debugger: When the user starts or stops the debugger, or when it hits a breakpoint.

Documents: When the user opens, edits, saves, or closes a document, e.g. a source file, we capture which of these actions she performed and the document's name. To reduce the number of edit events, we aggregate edits to the same document if less than two seconds pass between them. This results in one aggregated event whose duration is the time that elapsed between the first and the last aggregated edit.

Navigation: When the user navigates the code base, e.g., when she `ctrl`-clicks in the source, we capture the code context and what she clicked on.

Test Runs: When the user runs tests, we capture for each test how long it took and weather it passed.

System: When the user locks the screen, the screensaver activates, the machine hibernates, or the machine wakes from any of these states, we capture a respective event.

Version Control: When the user uses version control, e.g., when she commits her changes, we capture the respective action. By monitoring Git's ref-log, we do this even if the user uses an IDE-external tool.

Windows: When the user opens, closes, or switches between windows, we capture the window's title and which of these actions she performed. We do the same for Visual Studio's main window to identify when the user started, left from, returned to, or closed the IDE.

B. Event Manager

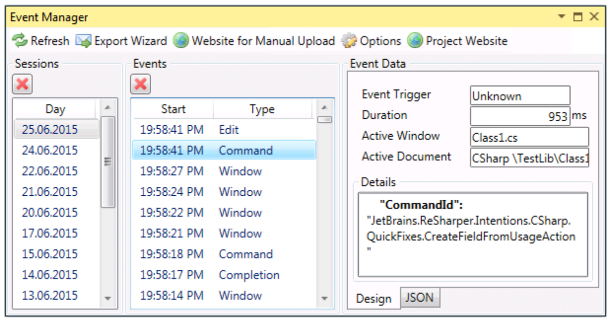
A developer's interactions with her IDE reveal sensitive information, such as work times or tasks. To encourage participation, we want developers to comprehend and control what FEEDBAG collects. Therefore, the *Event Manager* allows them to view and delete, but not alter, stored events via the UI shown in Figure 1. Though deletion is a threat to validity, we deem it very unlikely that developers systematically delete individual events in the huge event stream. They are more likely to delete entire days, which is equivalent to simply deactivating FEEDBAG for that day. The Event Manager displays the interactions recorded by the Event Generator in realtime. This allows us to demonstrate FEEDBAG's activity, even though it generally works transparent to the user.

Events are stored locally on the developer's machine. She is regularly informed about how many events were generated and asked to upload them. This simple guided process is the only user action FEEDBAG requires. Once the developer decides to provide her data, she has two options, shown in Figure 2: (1) Upload it directly to our Server or (2) export it to a ZIP archive and upload it via a web form. Either way, we ultimately receive the same data. FEEDBAG provides the second option only to allow for a manual review before submission.

On upload, the Event Manager asks the developer to optionally provide demographical information about herself (Figure 3). If she decides to do so, this data helps us characterize the population of contributors. The information is added to the upload as an additional event of the special type *User Profile*.

The interaction events themselves contain no information that allows one to directly link them back to the contributor. However, assuming that she commits her work to a public source-code repository, it is feasible to establish such a connection. To control what they publish about themselves, developers may remove (1) timestamps, (2) durations, (3) identifiers of project-specific methods and classes, or (4) any combination of these from the interaction events on submission.

After the export, the Event Manager deletes all local events.



Day
25.06.2015
24.06.2015
22.06.2015
21.06.2015
20.06.2015
17.06.2015
15.06.2015
14.06.2015
13.06.2015

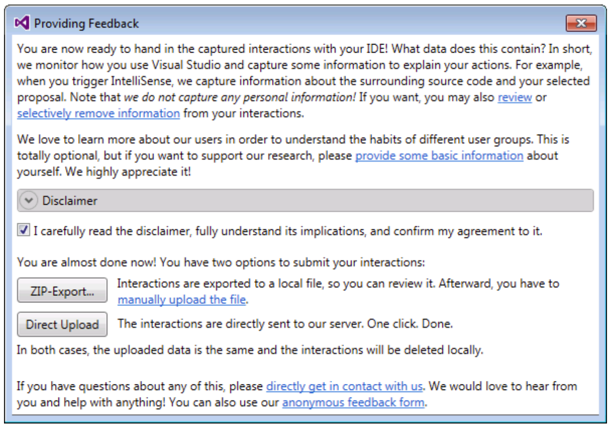
Start	Type
19:58:41 PM	Edit
19:58:41 PM	Command
19:58:27 PM	Window
19:58:24 PM	Window
19:58:22 PM	Window
19:58:21 PM	Window
19:58:18 PM	Command
19:58:17 PM	Completion
19:58:14 PM	Window

Event Trigger: Unknown
Duration: 953 ms
Active Window: Class1.cs
Active Document: CSharp \TestLib\Class1

Details

```
    "CommandId":
"JetBrains.ReSharper.Intentions.CSharp.
QuickFixes.CreateFieldFromUsageAction
"
```

Figure 1. Screenshot of the Event Manager



You are now ready to hand in the captured interactions with your IDE! What data does this contain? In short, we monitor how you use Visual Studio and capture some information to explain your actions. For example, when you trigger IntelliSense, we capture information about the surrounding source code and your selected proposal. Note that *we do not capture any personal information!* If you want, you may also review or selectively remove information from your interactions.

We love to learn more about our users in order to understand the habits of different user groups. This is totally optional, but if you want to support our research, please provide some basic information about yourself. We highly appreciate it!

☑ I carefully read the disclaimer, fully understand its implications, and confirm my agreement to it.

You are almost done now! You have two options to submit your interactions:

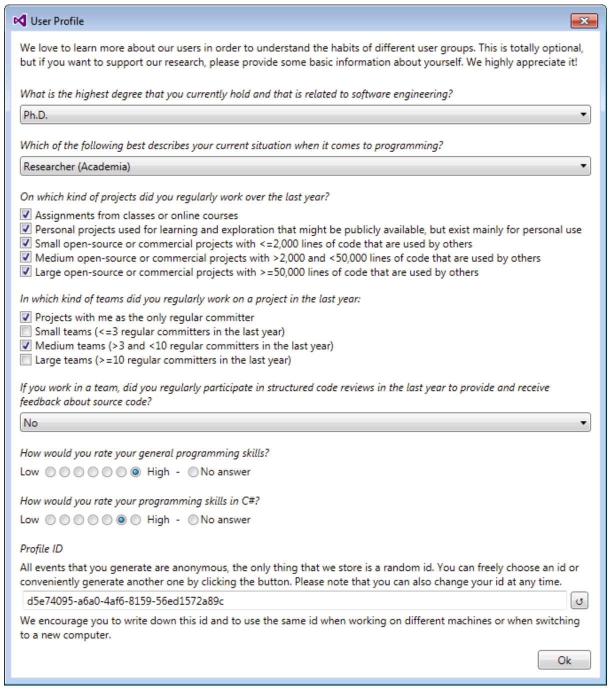
ZIP-Export... Interactions are exported to a local file, so you can review it. Afterward, you have to manually upload the file.

Direct Upload The interactions are directly sent to our server. One click. Done.

In both cases, the uploaded data is the same and the interactions will be deleted locally.

If you have questions about any of this, please directly get in contact with us. We would love to hear from you and help with anything! You can also use our anonymous feedback form.

Figure 2. Upload Dialog



We love to learn more about our users in order to understand the habits of different user groups. This is totally optional, but if you want to support our research, please provide some basic information about yourself. We highly appreciate it!

What is the highest degree that you currently hold and that is related to software engineering?
Ph.D.

Which of the following best describes your current situation when it comes to programming?
Researcher (Academia)

On which kind of projects did you regularly work over the last year?
☑ Assignments from classes or online courses
☑ Personal projects used for learning and exploration that might be publicly available, but exist mainly for personal use
☑ Small open-source or commercial projects with <=2,000 lines of code that are used by others
☑ Medium open-source or commercial projects with >2,000 and <50,000 lines of code that are used by others
☑ Large open-source or commercial projects with >=50,000 lines of code that are used by others

In which kind of teams did you regularly work on a project in the last year:
☑ Projects with me as the only regular committer
☐ Small teams (<=3 regular committers in the last year)
☑ Medium teams (>3 and <10 regular committers in the last year)
☐ Large teams (>=10 regular committers in the last year)

If you work in a team, did you regularly participate in structured code reviews in the last year to provide and receive feedback about source code?
No

How would you rate your general programming skills?
Low ○ ○ ○ ○ ○ ○ ● High - ○ No answer

How would you rate your programming skills in C#?
Low ○ ○ ○ ○ ○ ● ○ High - ○ No answer

Profile ID
All events that you generate are anonymous, the only thing that we store is a random id. You can freely choose an id or conveniently generate another one by clicking the button. Please note that you can also change your id at any time.
d5e74095-a6a0-4af6-8159-56ed1572a89c
We encourage you to write down this id and to use the same id when working on different machines or when switching to a new computer.

Figure 3. User-Profile Dialog

C. Server

The Server accepts interaction-event archives directly from the Event Manager or via manual upload. It checks that the received file complies to the data format exported by the Event Manager and stores it on server-side. The Server does not log any information about the sender.

III. Application of FeedBaG

In this section, we describe how we previously used FeedBaG and discuss potential future uses.

A. Previous Applications

We recently used FeedBaG to conduct a large-scale study, recording 6,355 hours of work time at the development department of an industry partner [5]. In this study, we gained valuable insights into the activities developers perform during an average work day and the tools they use. For example, we found that our participants navigate their codebase for 22.4% of the time they spend in their IDE. To this end, they open files through the project tree and use both general purpose and code-specific searches frequently. We observe that navigation time strongly correlates with short inactivities, which likely correspond to periods developers use for code understanding. Thus, we hypothesize that the amount of navigation is an indicator for the need for code comprehension. We find that this observation differs from a previous study on the Pharo IDE [3]. Future work could investigate the causes of these differences to identify opportunities to improve IDEs.

B. Applications in Future Work

We recently extended FeedBaG to collect more detailed information about code-comprehension activities. With this new version, we are currently building a dataset of IDE usage by different types of developers (e.g., academic and industrial) to investigate differences between these groups. We will publish this dataset for others to use it as well.

To reveal issues and opportunities for code-comprehension assistance in IDEs, it is important to compare IDE interactions from different settings. Therefore, we publish FeedBaG and encourage others to create additional public datasets with it.

In our ongoing work, we use the document-editing and code-completion events to learn how developers explore and use APIs. The interactions give us an oracle to evaluate respective tool assistance, such as code completion. We believe that other recommender systems for software engineering can be evaluated in a similar fashion, using FeedBaG datasets.

FeedBaG enables further empirical research, like studies on tool adoption. For example, researchers could replicate the study on unit testing that Beller et al. [4] conducted, to compare unit testing between Eclipse and Visual Studio.

Furthermore, we believe that FeedBaG opens up new opportunities to investigate the differences in IDE usage between different programming languages. Since the .NET environment encourages multi-language projects, one could analyze differences in the behavior of the same developer when switching between languages.

IV. Related Tools

In the last years, researchers implemented multiple instrumentations of IDEs. The one most similar to ours is a tool called Blaze [8]. Blaze, like FeedBaG, instruments the Visual Studio IDE to record developers' interactions. Unlike our tool, Blaze provides game-like feedback to encourage the usage of certain tools and practices. Neither the tool nor the respective datasets are publicly available. Snipes et al. [9] recently published a guide on how to instrument IDEs, based on their experience with developing Blaze. Unfortunately, the guide became available only after we conducted our study. Nevertheless, we find it to be a valuable resource and we can generally confirm their findings from our own experience.

Minelli et al. [3] developed an instrumentation for the Pharo IDE. They conducted a study very similar to ours, which enabled us to make many interesting observations about the differences between Pharo and Visual Studio [5]. Kersten and Murphy [10] developed Mylyn, an instrumentation for the Eclipse IDE. Their interaction events are more coarse grained than ours. Also Mylyn comes with an assistance tool that helps developers focus on their current task context, whereas FeedBaG is independent of other tools.

V. Summary

We present FeedBaG, an interaction tracker for Visual Studio. We used FeedBaG in a large-scale study, identifying code-comprehension activities, amongst other things. Future work should explore interactions from various settings, to identify issues and opportunities of IDE assistance tools.

VI. Acknowledgements

We thank our students D. Albrecht, A. Bauer, U. Fahrer, M. Kämmerer, S. Kemper, F. Weirich, and M. Zimmermann for their work on FeedBaG. The work presented in this paper was partially funded by the German Federal Ministry of Education and Research (BMBF), grant no. 01IS12054. The authors assume responsibility for the content.

References

[1] C. Parnin and C. Görg, "Building Usage Contexts During Program Comprehension," ICPC '06, 2006, pp. 13–22.

[2] G. Murphy, M. Kersten, and L. Findlater, "How Are Java Software Developers Using the Elipse IDE?" *IEEE Software*, vol. 23, no. 4, 2006.

[3] R. Minelli, A. Mocci, and M. Lanza, "I Know What You Did Last Summer," ICPC '15, 2015, pp. 25–35.

[4] M. Beller, G. Gousios, A. Panichella, and A. Zaidman, "When, How, and Why Developers (Do Not) Test in Their IDEs," ESEC/FSE '15, 2015, pp. 179–190.

[5] S. Amann, S. Proksch, S. Nadi, and M. Mezini, "A Study of Visual Studio Usage in Practice," SANER '16, 2016.

[6] "FeedBaG: An Interaction Tracker for Visual Studio – Online Artifact," http://www.st.informatik.tu-darmstadt.de/artifacts/feedbag/.

[7] "ReSharper," https://www.jetbrains.com/resharper/. Last checked on November 13, 2015.

[8] W. Snipes, A. R. Nair, and E. Murphy-Hill, "Experiences Gamifying Developer Adoption of Practices and Tools," ICSE '14, 2014, pp. 105–114.

[9] W. Snipes, E. Murphy-Hill, T. Fritz, M. Vakilian, K. Damevski, A. Nair, and D. Shepherd, *A Practical Guide to Analyzing IDE Usage Data*. Elsevier Inc., 2015, pp. 85–138.

[10] M. Kersten and G. C. Murphy, "Using Task Context to Improve Programmer Productivity," FSE '14. ACM, 2006, pp. 1–11.