Interactive Visualization of Developer‘s Actions

Lukáš SEKERÁK*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
sekerak.luk@gmail.com

Abstract. Software product development provokes numerous unanswered questions which may not be easily resolved. Many visualization techniques do not offer the option to view the development process from the perspective of particular steps of the developer. Step-by-step approach enables visualisation of various actions taken during the pro-cess – such as utilization of a browser or communication/information channels for trouble-shooting etc. Another significant indicator to be mindful of is the transcription of own or externally acquired code. These actions are parts of software product development and their visualization may give answer to questions.

1 Introduction

In the development of software products there are many unanswered questions, which cannot always be easily answered. Many visualization techniques do not offer the opportunity to view the development of software product in terms of developer’s behaviour. Which is based on collected developer actions. Through the visualization of these actions, we could identify different patterns of developer’s behaviour at his work. We could identify interesting events or anomalies that have detrimental impact on development.

Actions that we may be interested in would include, for example, use of a browser, use a of communication tools for problems solving. Further development environment and etc. Because of the fact that, the developer uses many tools in his work, we will focus only on some certain actions. We will be interested in actions such as, when the developer activate the browser. Did he open it just, when he had implemented the functionality into the system, or did he copy something from a website, or did he just read the documentation from portal?

The sequence of actions consists of developer activity. As result of the analysis of work activities of the user (developer), we obtain user model as a set of user characteristics and his behaviour over the time and in the working environment. Work activity is a set of user actions for a certain period, in the context of data he is working with, the working environment, respectively tool equipment (hardware and software) computer. The activity can be understood as a demarcation of the two major events, in which the work activity acquires logical meaning.

[*] Master degree study programme in field: Information Systems
Supervisor: Dr. Alena Kovárová, Faculty of Informatics and Information Technologies STU in Bratislava

2 Related work

In this section we will have a look at the different views of software development. These views watch / monitor certain features of the software project. These features are then measured and assessed.

1. Tracking code structure
2. Tracking source code metrics
3. Tracking tasks (Gantt diagram)[8]

We analysed several tools in visualization of development area. For example: Source Miner Evolution (SME)[9], YARN[7], Forest Metaphor[3], ClonEvol[6], Gevol[2], EPOSee[1]. We found, that these tools are focused only on the code structure or metrics. Often these views were combined. However, no tool does not focuses on developer's actions or his behaviour during the development. The measured information are visualized in animation using the time line.

The analysis of work "Using developer activity data to enhance awareness during collaborative software development" by Ferguson et al. provides a good overview of 12 tools [10], for different purposes and dedicated to the developer's activities. For example, Team Tracks, the visualization tool, that displays the current activity of the developer. It display which files are currently edited and who altered them. Accessory to the IDE environment monitors which files were most frequently visited and assumes that these files can be problematic. The tool is designed for teams and management of collaborative work. None of the mentioned tools dealt with visualization of software development by a view similar to ours. Article indicates the importance of capturing data for tracking developer's activities and of course the visualization of this data.

3 Software development visualization method

Our method is based on developer's actions. These actions can be tracked by several tools, we have chosen PerConIK project. Shortly, we will describe this project, types of events and how we will build activity from these events.

3.1 PerConIK data source

PerConIK (*Personalized Conveying of Information and Knowledge, per-conik.fiit.stuba.sk*[4]). Part of the project is also a solution that records and analyses user work activity, in order to analyse behaviour of the user in the context of used software tools, that are used, but also in the context of data (documents, source code, Web, ...), he works with. The solution collects data on each keystroke, mouse click, browser opening and many others. Indeed, these data are linked to version control system.[5]

3.2 Activity tracker

It is an application that monitors the activities of the developer, as a user of the operating system. It is the central application in the operating system, which collects data also from other environments with additional components. These components follow document editing, web browser etc. In general, it monitors the currently running processes, open windows and the users' interaction. Tracker monitors activities also in the integrated developer environment (monitor the operations over source code).

3.3 Building activity from actions

When the developer's activity is created, it has two major events, such as the opening and the closing of tab in the browser. During this time period, the developer could look at the website, or maybe not have to. If the developer opened tab and closed it in a short period of time, it is highly probable that the developer was on the web site. This time period will be defined by *threshold* and this will determine the creation of activities from events (actions). If an event of the same type occurs within a given time interval, we can say that the developer has worked on the activity, that is declared by the type of the event.

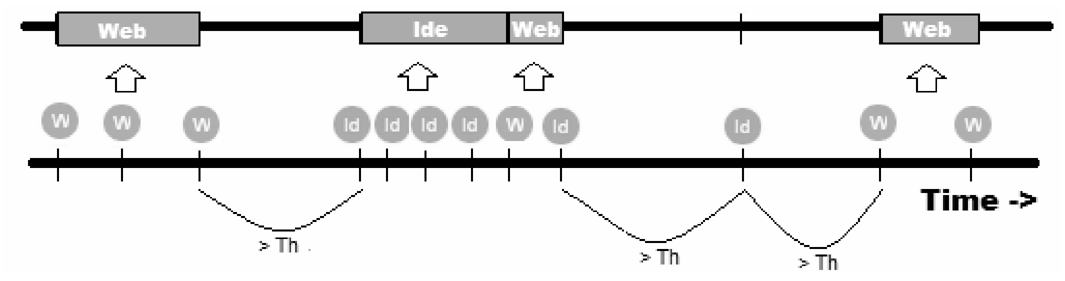

Figure 1. Building of activities from events. First line showing builded activites, second line events at time line. We use threshold value between events to determine, when event is part of activity and when it is not.

3.4 Privacy policy of developer

Captured developer's actions affect all processes of the developer's workstation. Including the web browser. Program also traces the web addresses visited by developer. However, these processes may interfere with the privacy of the developer.

4 Evaluation

4.1 IVDA tool

We have designed and implemented a tool to analyse features of activities and to visualize them. Individual features are visualized with the aid of specific graphs, focusing on data type. The user creates graphs and visually monitor the information. Tool support several types of graphs:

1. Duration of the time spent on web portals
2. Graph of source code changes (specifies when changes occur and how big they were)
3. Graph of picked features for activities (specify when activities occur, how much time did they take and shows metric determined by the type of activity)
4. Graph showing distribution of events (like a histogram)
5. Graph showing distribution of activities (focuses only on duration of the activity)
6. Graph showing changes over files and projects
7. Count of visited domains
8. Detail of run and used processes at developer's workstation
9. Graph showing grouped information for activities per some period.
10. Timeline visualization showing detail of events and measured metrics

Each graph is generated as the user has requested with the help of interactive toolbar. Graphs are generated for specified developer and time period. User can interact with graph too, can set another period or see some details about events and activities.

4.2 Graph of activities and their metrics

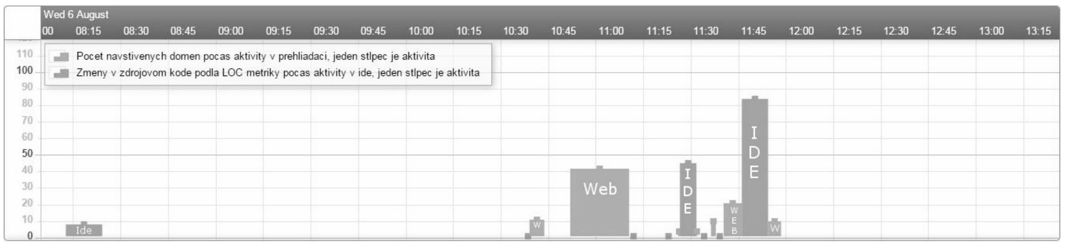

Figure 2. Graph showing developer activities at 6. August. Activities are tracked at Web and IDE enviroment.

Figure 2 shows the developer's activities within the browser and in the IDE. We have chosen the developer called Mouse, and the time period from 8:00 to 13:20, 6 August. From the graph, modified histogram, we can see what time the activities occurred. By looking at the width of the columns, we can see how long they lasted. Furthermore, we can see the characteristic of activity, according to the height of the column. For example, for the activity in the browser, we count the number of domains visited. For IDE activity, we count how many lines has changed within the activity.

Using this modified histogram, we can deduce that at 8:15 he devoted 10 minutes at Ide. Where he made the changes in the source code, and has changed 8 lines. Between 8:25 and 10:30, developer worked with something other than the Web or IDE, or has been off-line.

The developer visited the most of the domains between 10:45 and 11:10, when he visited about 40 domains. It is possible for the developer to visit such high number of domains in such a short time, for example, when dealing with a problem. At the time of 11:45 developer made a change or many changes, when he changed about 80 lines of code in 10 minutes. These changes may have been made, for example, by commenting 80 lines of source code or by refactoring. Next graphs bring next details about this change. After 12:00, the developer did not made any activity. Histogram mainly refers to the duration of activities and their order.

4.3 Hypothesis result

We defined our main aim, hypothesis. "If a developer uses a browser (a particular portal) frequently during code-writing, how often is the code rewritten? If so, does this apply to every other developer in the same manner?" One of our graphs attempts to answer to this hypothesis. We believe that this graph will have bigger signification with more data.

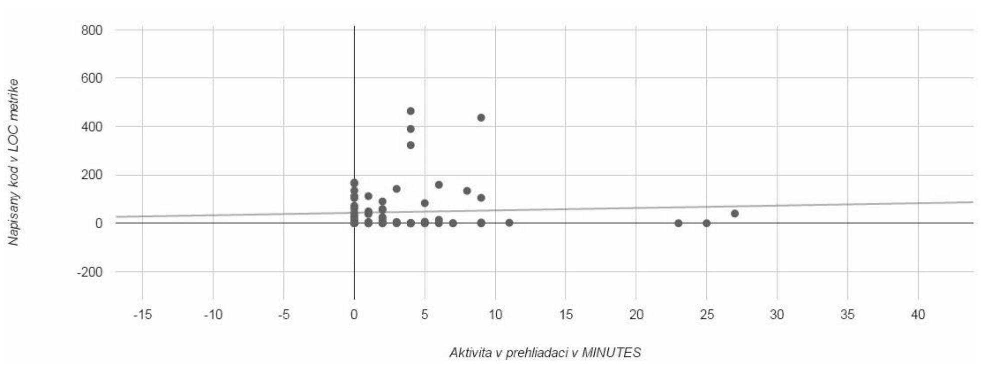

Figure 3. Scatter graph. Every point represent ratio between measured time of the Web activity (in minutes) and changes at source code (by LOC metric) in the next corresponded IDE activity. Corresponded IDE activity is activity, which happen immediately after the previous (Web) activity.

Actually user can generate another interested graphs to insight into developer's work.

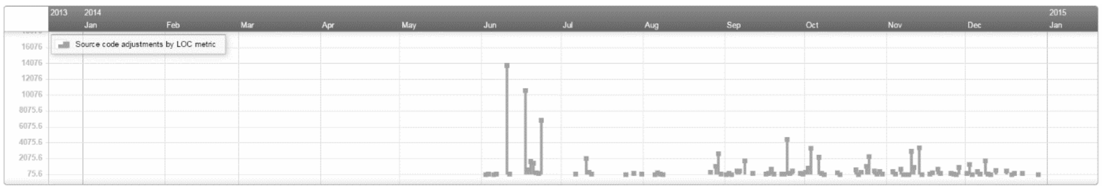

Figure 4. Timeline view of source code adjustment by specific developer (in LOC metric) on year scale.

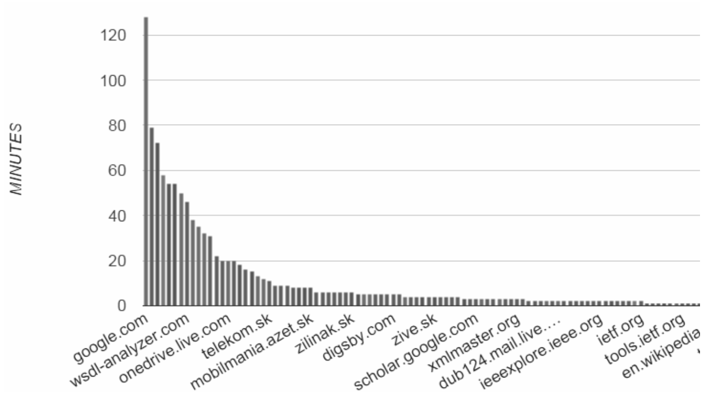

Figure 5.Visited domains and sum of time spent on them.
Data were tracked in one month of developer's work.

5 Conclusions

At the beginning of this paper we pointed out that a lot of visualization tools do not use view of developer's behaviour to visualize the development of the software. We think that this view has big potential and can be used to find answers to various interested questions. To support this thought, we have created generic visualization, where user can generate graphs. Visualization show information about activates and developer's actions and their metrics. In this paper we have showed only two graphs, although many graphs can be generated. By some changes we could create any graphs which will answer any software development questions based on developer's actions.

Our solution is based on restful service and visualization. Service is reusable for any *PerConIK* project. Visualization is web-based and can be easily part of another tool, such as a tool for managing project. Any manager may use this tool to identify anomaly and trends of their developers. For example, he can find out if using some particular portal has a big impact on developer work. Also this tool can help any scientist to answer hypothesis in software development area.

We have shown that this new view of software development is important and has the potential to be developed further. The aim was to create a tool, visualization, which would respond to the hypothesis which the other instruments could not answer. The scatter graph shows no direct answer whether the hypothesis is true or not. This may be due to lack of data. The next graphs dealt with another hypothesis, they provide better quality of view at the developer's activities.

Acknowledgement: This Software is the partial result of the Research and Development Operational Programme for the project Research of methods for acquisition, analysis and personalized conveying of information and knowledge, ITMS 26240220039, co-funded by the ERDF.

References

[1] BURCH, M. et al. EPOSee: A Tool For Visualizing Software Evolution. In *3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis* [online]. 2005. s. 1–2.

[2] COLLBERG, C. et al. A system for graph-based visualization of the evolution of software. In *Proceedings of the 2003 ACM symposium on Software visualization - SoftVis '03* . New York, New York, USA: ACM Press, 2003. s. 77.

[3] ERRA, U. et al. Visualizing the Evolution of Software Systems Using the Forest Metaphor. In *2012 16th International Conference on Information Visualisation* . 2012. s. 87–92..

[4] GRATEX PerConIK. [Online, accessed December 12, 2014]. Available at: <http://perconik.fiit.stuba.sk/>.

[5] GRATEX PerConIK : Používateľská príručka. [Online, accessed December 12, 2014]. Available at: <http://perconik.fiit.stuba.sk/UserActivity/Default/UserManual>.

[6] HANJALIC, A. ClonEvol: Visualizing software evolution with code clones. In *Software Visualization (VISSOFT), 2013 First IEEE ...* 2013. s. 1–4.

[7] HINDLE, A. et al. YARN: Animating Software Evolution. In *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*. 2007. s. 129–136.

[8] MALETIC, J.I. et al. A task oriented view of software visualization. In Proceedings First International Workshop on Visualizing Software for Understanding and Analysis . 2002. .

[9] NOVAIS, R. - LIMA, C. An interactive differential and temporal approach to visually analyze software evolution. In *Visualizing Software for Understanding and Analysis*. 2011. s. 1–4.

[10] ROEHM, T. - MAALEJ, W. Automatically detecting developer activities and problems in software development work. In Proceedings - International Conference on Software Engineering . 2012. s. 1261–1264.