

An Approach to Quantify the Productivity of Software Developers towards the Perceived Productivity

J Rajeshwar

CSE Department, Guru Nanak Institutions Technical Campus (Autonomous), Hyderabad, Telangana, India

*Corresponding Author: prof.rajeshwar@gmail.com

Available online at: www.ijcseonline.org

Accepted: 26/Jan/2019, Published:31/Jan/2019

Abstract -- Many software improvement agencies attempt to beautify the productivity of their builders. All too regularly, efforts geared towards increasing developer productiveness are undertaken without a proper knowledge of how exactly builders spend their time at their work and how it impacts their own belief of productivity. Verifying earlier findings, we try to found that developers pay their time on a good type of tasks and switch frequently among hem, succeeding in particularly fragmented work. Our findings enlarge past existing studies therein we tend to correlate builders' work conduct with perceived fecundity. Although productiveness is based on individuals, developers may be roughly gathered in morning sessions, low at lunch and afternoon. A continuous linear regression per participant found that greater grade persons usually use a high-quality, and emails, deliberate meetings and unrelated web sites with a terrible belief of productivity. We discuss opportunities of our findings, the capability to expect high and occasional productiveness and endorse layout tactics to create higher tool guide for planning builders' workdays and enhancing their work productivity.

Keywords— quantification, perceived productivity, regression

I. INTRODUCTION

A software developer's work day may be well influenced by a wide variety of factors such as the tasks being performed, meetings, and interruptions from co-workers, the infrastructure or the workplace environment. Number of these factors end in activity and context switches that may cause fragmented work and can have a negative impact on the developer's perceived productivity, progress on tasks, and quality of output. As a result, researchers and practitioners both have a long interest in better understanding how developers work and how their work could be quantified to optimize productivity and efficiency. Researchers have investigated work practices and work fragmentation in detail from various perspectives, specifically the effect of interruptions on fragmentation and how developers organize their work in terms of tasks and working spheres. Using both a diary and an observational study format to understand software developer work practices, Perry and colleagues gained several insights, including that most time was spent coding, and that there was a substantial amount of unplanned interaction with colleagues. Singer and colleagues, using several study methods including tool usage statistics, found that developers spent most of their time reading documentation and that search tools were the most heavily used. Since the time these earlier studies on developers' work practices were conducted, empirical studies of software

development have focused more on particular aspects of a developer's work day.

In this paper, we study developers' work practices and the relationship to the developers' perceptions of productivity more holistically, while also examining individual differences. In particular, our study seeks to answer the following research questions:

Q1 How a developer's work day look like?

Q2 how fragmented may be a developer's work?

Q3 are there any noticeable trends in how developers understand their productivity?

Q4 what is the connection between developers' activity and perceived productivity at work?

To examine these queries, we planned and organized a study involving the observance of twenty developers' interactions with their laptop over a 2 week time period. From this observance, we tend to were able to gather logs describing however a developer was interacting with the pc (i.e., through the keyboard or the mouse) and in what applications the interaction was occurring. Our observance also gathered self-reports from the developers concerning their current

task(s) at hour time intervals, and a self-rating of their perceived productivity. The twenty developers from whom we tend to gather knowledge worked for four totally different corporations of variable size, with variable comes, project stages and customers, providing additional diversity in our results .

The next step is to research the work in more details: excluding conferences, developers stay solely between .3 to 2.0 minutes in associate activity before switching to a different activity. These terrible short times per activity and therefore the kind of activities a developer pursues daily illustrate the high fragmentation of a developer's work. From participant's self-reported, perceived productivity we found that though there was lots of deviations between people, these may be categorized into 3 groups: morning individuals, afternoon individuals, and people whose perceived productivity swaybacked [12] at lunch. Morning individuals typically return to figure a trifle bit earlier, and acquire the foremost necessary things done before the group arrives. Afternoon folk sometimes arrive later and pay most of their time with conferences and emails, and acquire stuff drained in the afternoon, so feeling a lot of productive [11]. These results counsel that whereas data staff normally has various perceived productivity patterns, people do seem to follow their own habitual patterns for every day.

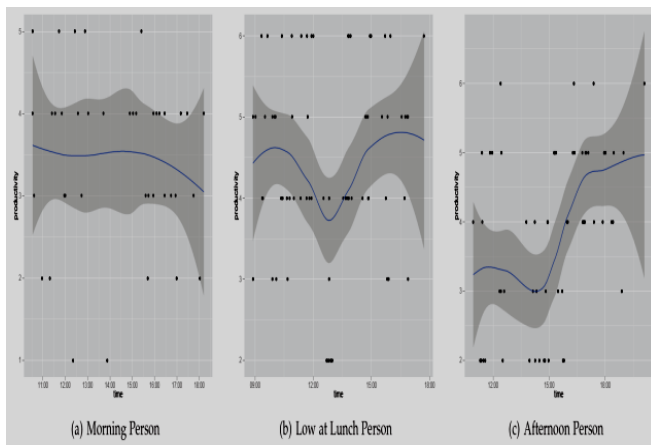


Figure 1: Production Level at Different Sessions

II. LITERATURE SURVEY

T. DeMarco and T. Lister et.al [1] proposed that Wide variation in programmer performance has been frequently reported in the literature. In the absence of alternative clarification, managers have to just accept that the variation is attributable to individual characteristics. J. Singer, T. Lethbridge et.al [2] presents work practice data of the daily activities of software engineers. Four separate studies are presented; one looking longitudinally at an individual SE; two looking at a software engineering group; and one

looking at company-wide tool usage statistics. D. E. Perry, N. A. Staudenmayer et.al [3] determined however technology affects the software development method, researchers often overlook structure and social issues [3]. M. Zhou and A. Mockuset.al [5] [6] proposed Outsourcing and off-shoring lead to a rapid influx of new developers in software projects. That, in turn, manifests in lower productivity and project delays. When adjusted for the task difficulty, developer productivity did not plateau but continued to increase over the entire three year measurement interval [13] [14]. Based on the literature review it was found that software productivity measurement can be done using SLOC (Source Lines of Code), function points, use case points, object points, and feature points [4]. A. N. Meyer, T. Fritz et.al [7] explained better the software development community becomes at creating software, the more software the world seems to demand. M. Czerwinski, E. Horvitz, and S. Wilhite et.al [8] reported on a diary study of the activities of information workers aimed at characterizing how people interleave multiple tasks amidst interruptions. C. Parnin and S. Rugaberet.al [9] explained Interrupted and blocked tasks are a daily reality for professional programmers. Unfortunately, [10] the strategies programmers use to recover lost knowledge and rebuild context when resuming work have not yet been well studied.

III. PROPOSED SYSTEM

Understanding developer productivity is important to deliver software on time and at reasonable cost. Yet, there are numerous definitions of productivity and, as previous research found, productivity means different things to different developers. Through a cluster analysis, we tend to determine and describe six teams of developers with similar perceptions of productivity: social, lone, focused, balanced, leading, and goal-oriented developers. We argue why personalized recommendations for software developers' work is vital and discuss style implications of those clusters for tools to support developers' productivity

SME Algorithm (Structure Mapping Engine)

In analogy, a given state of affairs is known by comparison with another similar state of affairs. Analogy is also wont to guide reasoning, to get conjectures concerning AN unknown domain, or to generalize many experiences into AN abstract schema. Consequently, analogy is of nice interest to each psychological feature psychologists and computing researchers. Psychologists would like to clarify the mechanisms underlying analogy so as to grasp human learning and reasoning. Artificial Intelligence researchers would like to emulate figurative process on computers to supply additional versatile reasoning and learning systems..It constructs all consistent ways that to interpret a possible analogy and will thus while not backtracking. SME provides a "tool kit" for building matchers satisfying the structural

consistency constraint of Gentner's theory. The remainder of the constraints defining a intermediary are such by a group of rules, that indicate native, partial matches and estimate however powerfully they ought to be believed. The program uses these estimates and a unique procedure for combining the native matches to with efficiency turn out and appraise all consistent international matches.

IV.IMPLEMENTATION

User Interface

In this we design the windows for the project. These windows are used for secure login for all users. To connect with server user must give their username and password then only they can able to connect the server. If the user already exists directly can login into the server else user must register their details such as username, password and Email id, into the server. Server will create the account for the entire user to maintain upload and download rate. Name will be set as user id. Logging in is usually used to enter a specific page.

Create and Assign

In this module admin can login and he/she will create the employee with basic data. And that data is given to the employee. Here admin will create the projects. Work will assign to different teams with team leaders. And monitor the working of the employees and details about the projects that which are under processing.

Activity and Productivity Verification

After Admin Login then he/she will monitor the activities of particular employee. That means how many hours the employee is in active and inactive time . And how many times employee switch from one work to another work. Based on all these things the productivity level of the employee is monitored.

Work Life of Development

In this employee should login initially. Employee will see the project which is assigned by the admin. And then he will start the project and give the status of that project daily. And here they can switch from one task to another task.

Dashboard

In this employee can see his/her complete profile and they can edit their profile. They can chat or send the any files or images to their teammates. They can see the files which were send by other employee.If any Employee is having any queries or anything that is not related, then he can raise question to the Admin.

Survey Report

In this employee should login initially, then employee gives the self report on his/her corresponding project.Regarding the work the Employee has done Admin may raise questions after the report that is send by him. Some questions will display here for those questions the employee will provide some answers. Based on the answers the status of the employee will be calculated.

V.RESULTS



Figure 2: Home Page

Figure 2 illustrates the home page of the application. In this page there are three tabs. First tab is for Manager login, Second is for Employee login, Third is for Client Login.

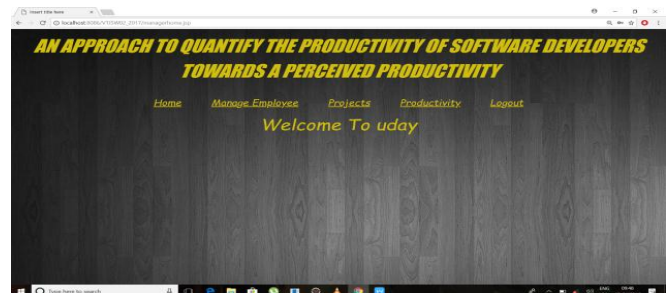


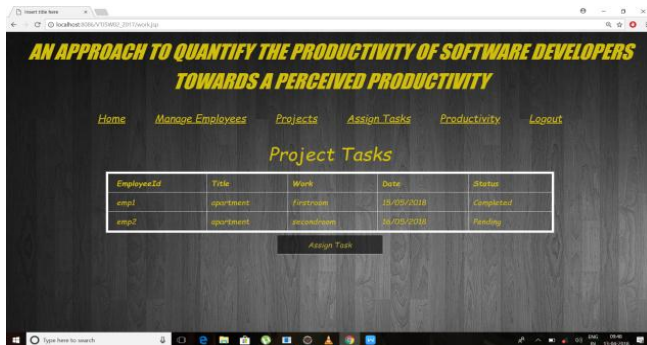
Figure 3: Mangers' Home Page

Figure 3 illustrates the home page of the Manager. In this page the first tab is to manage employee i.e. he can add employees required for the project, second tab is for checking the project status, third tab is for productivity level and last tab is logout tab.



Figure 4: Project Details

Figure 4 illustrates the project details. Manger directly gets the projects that are required by the client. After getting request from the client Manager has to accept the project then it will be treated under the ongoing project.



Project Tasks

EmployeeId	Title	Work	Date	Status
emp1	apartment	Interior	05/05/2018	Completed
emp2	apartment	interiorwork	06/05/2018	Pending

Assign Task

Figure 5: Status Report

Figure 5 illustrates the status report of the ongoing project. Manger will check under the assigned tasks tab. So that he will get the status of the each employee whether it is pending or completed. Employee will send the report to the Manager after completion of the task that is assigned to him.



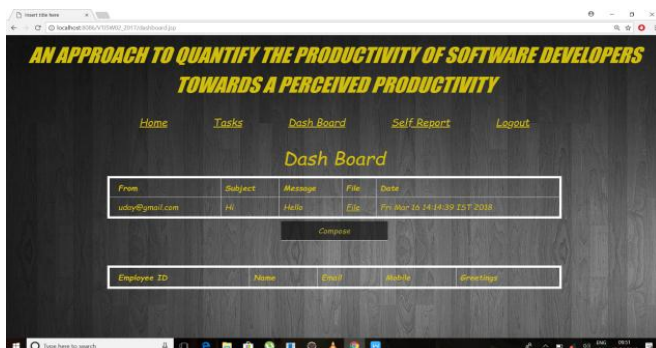
Production Level

UserID	Number of Switchings	Active Time	Inactive Time	Mouse Clicks	Key Strokes
harsh24@gmail.com	4	0:00:00:00:00	0:00:00:00:00	5	11

Summary Report

Figure 6: Productivity Level

Figure 6 illustrates the productivity level of the project. In this page Manager checks the number of switching done by the employee, while shifting from tasks Manager observes the search results whether it is related to the task or personal work, active and inactive time of the employee. And also the number of mouse clicks and the Key strokes made by the employees. Thus increase in better productivity level.



Dash Board

From	Subject	Message	File	Date
uday@gmail.com	Hi	Hello	Epic	Fri, Mar 16, 14:14:39 +10'30'00

Compose

Employee ID	Name	Email	Mobile	Greeting

Figure 7: Dashboard

Figure 7 illustrates the dashboard. In this page employee can see his profile. Employee can communicate with the others if he has any queries regarding the project or any personal work like chatting, sending mails, greetings also. This is considered as the inactive time thus considered as decrease in the productivity level.

VI. CONCLUSION

As a result, researchers and practitioners both have long interest in better understanding how developers work and how their work could be quantified to optimize productivity and efficiency. So this will increase the productivity levels and would result in good product in minimum time with better efficiency

REFERENCES

- [1] T. DeMarco and T. Lister, "Programmer performance and the effects of the workplace," in Proceedings of the 8th international conference on Software engineering. IEEE Computer Society Press, 1985, pp. 268–272.
- [2] D. E. Perry, N. A. Staudenmayer, and L. G. Votta, "People, organizations, and process improvement," IEEE Software, vol. 11, no. 4, pp. 36–45, 1994.
- [3] J. Singer, T. Lethbridge, N. Vinson, and N. Anquetil, "An examination of software engineering work practices," in CASCON First Decade High Impact Papers, ser. CASCON '10. IBM Corporation, 2010, pp. 174–188.
- [4] B. W. Boehm, "Improving software productivity," vol. 20, no. 9. IEEE, 1987, pp. 43–57.
- [5] A. N. Meyer, T. Fritz, G. C. Murphy, and T. Zimmermann, "Software developers' perceptions of productivity," in Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, ser. FSE 2014. ACM, 2014, pp. 19–29.
- [6] R. Van Solingen, E. Berghout, and F. Van Latum, "Interrupts: just a minute never is," IEEE software, no. 5, pp. 97–103, 1998.
- [7] S. T. Iqbal and E. Horvitz, "Disruption and recovery of computing tasks: Field study, analysis, and directions," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '07. ACM, 2007, pp. 677–686.
- [8] M. Czerwinski, E. Horvitz, and S. Wilhite, "A diary study of task switching and interruptions," in Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 2004, pp. 175–182.
- [9] C. Parnin and S. Rugaber, "Resumption strategies for interrupted programming tasks," Software Quality Journal, vol. 19, no. 1, pp. 5–34, 2011.
- [10] V. M. Gonz'alez and G. Mark, "Constant, constant, multi-tasking craziness: Managing multiple working spheres," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '04. ACM, 2004, pp. 113–120.
- [11] A. J. Ko, R. DeLine, and G. Venolia, "Information needs in collocated software development teams," in Proceedings of the 29th International Conference on Software Engineering, ser. ICSE '07. IEEE Computer Society, 2007, pp. 344–353.
- [12] R. Minelli, A. Mocchi, and M. Lanza, "I Know What You Did Last Summer – An Investigation of How Developers Spend Their Time," Proceedings of ICPC 2015 (23rd IEEE International Conference on Program Comprehension), pp. 25–35, 2015.
- [13] S. Amann, S. Proksch, S. Nadi, and M. Mezini, "A study of visual studio usage in practice," in Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER '16), 2016.
- [14] M. Zhou and A. Mockus, "Developer fluency: Achieving true mastery in software projects," in Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering, ser. FSE '10. ACM, 2010, pp. 137–146.