Investigating Truck Factor in Real-world Systems: a Survival Analysis

Guilherme Avelino

Federal University of Minas Gerais, Brazil
Eindhoven University of Technology, Netherlands

Advisor: Marco Tulio Valente (UFMG, Brazil)
Internship Advisor: Alexander Serebrenik (TUe, Netherlands)

1 Introduction

In previous study [2] we propose a new approach to estimate the truck factor of software systems and we apply it to investigate knowledge concentration in popular systems extracted from GitHub. Our results indicated that knowledge concentration is a real problem in most of these systems (65 % have $TF \leq 2$). Additionally, we asked the developers which are the development practices they see as most effective to attenuate the impact of truck factor event in their systems. Many system's developers answered that as open source systems they have a great community of developers that could provide skilled developers able to replace the lost ones.

Indeed, most of these systems follow a so-called onion model [6]—the project development team is organized in concentric circles, where the most inner circle includes the core developers while the outer circles contains more passive contributors. These model also express the substantial difference in scale between the group sizes fulfilling each role. The core developers is composed for a really limited number of components, while the peripheral groups are gradually bigger.

While some studies demonstrate the existence of rules to promote peripheral developers to inner layers [4, 5], it's not clear what extent the system's community is capable to provide developers to overcome the lost of its most important members? Additionally, what are the characteristics of a healthy community of developers? Seeking to answer these questions, we propose to conduct a study that by mining data from Github repositories identifies systems where a truck factor events occurred and use these systems to investigate which characteristics of the development environment helped they overcome or not such event. We describe as a truck factor event a situation in which all top-developers in the truck factor list, as computed adopting the metric defined in [1], do not contribute with the system code anymore. We argue that by building an empirical body of knowledge on how these systems dealt with the

lost of important developers can help to get parameters about how to organize and manage the development community to mitigate the risks associated to knowledge concentration.

2 Proposed Work

Our main objective in this study is to investigate the occurrence of the lost of key developers in real-software systems and identify factors that helped these systems to overcome or not this problem. More specifically, we aim to answer research questions like: *How common is the lost of the most important developers in open-source systems? What is the survival rate of these systems when they have to deal with a truck factor like event? Which are the development environment characteristics that distinct the systems able to survive?*

To tackle such research questions we intend to perform the following activities:

- **Activity #1 - Selection of target subjects:** our previous work [2] suggests that truck factor like events are not uncommon. Even without applying any specific methodology, in a sample of 133 systems we identified three systems in which an truck factor event occurred. We could investigate in depth the same sample, but aiming to identify a bigger number of cases a better option is to conduct the study in a larger set of systems. Therefore, we plan to apply similar procedure to select important and relevant systems in GitHub, but starting from a larger set of systems.

- **Activity #2 - Identify truck factor events:** to identify truck factor events we plan to apply the following procedure to the systems in our dataset: (*i*) break the development history of the system in n time frames (t_1 to t_{n-1}) of d days each one; (*ii*) initialize the current time frame $ct = t_2$; (*iii*) compute the truck factor of the system considering the development history before c_t, also getting the list of developers responsible for this number; (*iv*) if these developers become *leavers* of the system until the end of c_t, stop the algorithm and report the system; (*v*) else, increment c_t and return to *step iii*. We plan to adopt the same definition for *leavers* and *joiners* described in [3].

- **Activity #3 - Investigate the development environment of the systems:** after identify the systems where a truck factor event occurred, we plan to conduct an investigation of the development environment aiming to identify factors that may have influenced they overcome or not such event. As starting point in our analysis, we plan to investigate characteristics such as: workload distribution, number of uncovered files, number of *joiners*, collaboration patterns, and developers profile. An static view is useful to characterize the system development, but we also plain to apply a historical analysis to verify the changes introduced by truck factor event and how the development environment evolved to deal with this problem.

- **Activity #4 - Survey with systems developers:** we plan to complement the empirical analysis by applying a survey with the systems' developers. This activity will help us to contrast our findings and conclusion with the developers perception.

- **Activity #5 - Summarize the study in a paper:** we also intend to start a paper for a major software engineering conference/journal about our research and results.

3 Schedule

Table 1 presents the schedule of the activities proposed for the sandwich internship.

Table 1: Schedule of activities

Activity	2017				
	May	**June**	**July**	**August**	**September**
Activity #1	✓	✓			
Activity #2		✓	✓		
Activity #3			✓	✓	
Activity #4				✓	✓
Activity #5					✓

References

[1] Guilherme Avelino, Leonardo Passos, André C. Hora, and Marco Tulio Valente. Code authorship: an empirical analysis of the linux kernel evolution. Submitted to SANER 2017.

[2] Guilherme Avelino, Leonardo Passos, André C. Hora, and Marco Tulio Valente. A novel approach for estimating truck factors. In *24th International Conference on Program Comprehension (ICPC)*, pages 1–10, 2016.

[3] Eleni Constantinou and Tom Mens. Socio-technical evolution of the ruby ecosystem in github. In *IEEE 24th International Conference on Software Analysis, Evolution and Reengineering, SANER 2017, Klagenfurt, Austria, February 20-24, 2017*, pages 34–44, 2017.

[4] Nicolas Ducheneaut. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work*, 14(4):323–368, 2005.

[5] Chris Jensen and Walt Scacchi. Role migration and advancement processes in ossd projects: A comparative case study. In *29th International Conference on Software Engineering (ICSE)*, pages 364–374, 2007.

[6] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. Evolution patterns of open-source software systems and communities. In *5th International Workshop on Principles of Software Evolution (IWPSE)*, pages 76 – 85, 2002.