Socio-technical Evolution and Migration in Software Ecosystems

Eleni Constantinou and Tom Mens
Software Engineering Lab, COMPLEXYS Research Institute
University of Mons, Belgium
first.last@umons.ac.be

I. Introduction

Open source software ecosystems are formed by software projects that are developed and evolve together in the same environment [1]. Considering that most open source projects today are no longer developed in isolation [2], the research community has recently shifted its attention to *software ecosystems* [3]. Examples of software ecosystems include package distributions for programming languages (e.g., CRAN, CPAN, RubyGems, npm, PyPI) or operating systems (e.g., Ubuntu, Debian), and mobile app stores. A significant part of many of these ecosystems is developed through GitHub, a portal for distributed versioning allowing developers to create and fork projects, link these projects via dependencies, and collaborate through a pull-based development process [4]–[6].

Developing software in such a way is an inherently *social* activity, involving the interaction, communication and collaboration of multiple contributors to the same project and across interdependent projects. It is also an inherently *technical* activity, involving the creation, generation and modification of a multitude of software artefacts such as source code, documentation, tests, metadata, and many more.

The socio-technical evolution dynamics of software ecosystems is an emerging research subject [7]–[9], but current studies mainly focus on *temporary* changes of the ecosystem. For example, an analysis of temporary changes will consider a contributor who becomes temporarily inactive but continues to contribute later on. In contrast, this paper focuses on *permanent* changes in the ecosystem and measures their effect on the ecosystem's evolution. For example, a contributor may decide to leave the ecosystem, causing turnover in the projects he is contributing to, and perhaps leading to abandoned projects or files. The degree of turnover and abandonment may be indicative of quality problems in projects or in the ecosystem as a whole.

II. Socio-technical evolution

We investigated the evolution of the Ruby ecosystem in GitHub and considered two levels of granularity, namely the global ecosystem and the evolution around individual Ruby projects. Blincoe et al. [2] observed that ecosystems in GitHub tend to revolve around central projects with many other projects depending on these central projects, forming star-like patterns. By analogy, we will considered multiple base projects in Ruby, and studied the co-evolution with the forks of these projects. Table I presents descriptive statistics of the Ruby ecosystem. The observed period of the Ruby ecosystem starts on October 29th 2007 and ends on September 3rd 2016.

TABLE I
Ecosystem descriptive statistics

	Base	Forks	Ecosystem
Projects	10,792	49,101	59,893
Contributors	42,206	34,317	55,924
Touched Files	681,539	191,016	712,300
Commits	2,638,097	887,030	3,525,127
LOC	389,930,604	77,510,268	467,440,872

Next, we present the results of the socio-technical evolution of the Ruby ecosystem and study the following research questions:

How does the ecosystem grow over time?

We found that the main development activity takes place in the base projects and that the number of base projects and forks present a significant increase, while after February 2014 a decrease is observed. After February 2014, both base projects and forks are created less often and the number of base projects becoming obsolete presents a large increase compared to the ecosystem's history.

In an effort to explain these observations, we gathered anecdotal evidence from developer blogs and Ruby's mailing lists. We found discussions about how can Ruby challenge some new technologies, like Node.js[1], and the need for Ruby to transform. Also, we gathered anecdotal evidence about migrations from Ruby at that period: moving to Node.js due to performance issues[2] [3] and companies like Twitter moving to other technologies to handle increased traffic [4].

To support this anecdotal evidence, we investigated the activity in the Ruby and JavaScript ecosystems for the same timespan by measuring the number of active base projects according to their commit activity. We found that until November 2011 both ecosystems have comparable growth, while afterwards the JavaScript ecosystem in GitHub presents a larger growth compared to the Ruby ecosystem.

[1] http://blade.nagaokaut.ac.jp/cgi-bin/vframe.rb/ruby/ruby-talk/411961?411838-412226

[2] http://ilikekillnerds.com/2015/02/is-ruby-on-rails-dying/

[3] http://blog.parse.com/learn/how-we-moved-our-api-from-ruby-to-go-and-saved-our-sanity/

[4] http://www.theregister.co.uk/2012/11/08/twitter_epic_traffic_saved_by_java/

How does the technical part of the ecosystem evolve?

As expected, we observed an increase in the number of actively developed source code files, where the bulk of the development activity takes place in the base projects. After February 2014, a drop in the number of new source code files is observed for the base projects, indicating either the contributors' focus on maintaining existing files or a reduction in development effort. The average turnover and abandonment corresponds to 47% and 35% respectively for the base projects confirming the ecosystem growth until February 2014, where the abandonment starts to exceed the turnover. The average FileTurnover and FileAbandonment of forks corresponds to 60% and 42% respectively. This is an expected outcome considering the pull-based development process. We also observe the increased abandonment from February 2014 onward.

How do the social modifications in the ecosystem impact the technical artefacts?

To measure the impact of contributors abandoning the Ruby ecosystem, we measure their diversity index in the project-contributor graph of the Ruby ecosystem, inspired by the work of Posnett et al [10]. Diversity measures are borrowed from ecology and when applied to bipartite graphs, they express the specialization of a given species with respect to the species in the other level [11]. We calculated the relative entropy (a.k.a. Kullback-Liebler divergence) [9], [10] to measure the specialization, and therefore assess the relative risk, of people abandoning the ecosystem. We found that although the bulk of leavers have low specialization, during the ecosystem's evolution contributors with increased specialization abandon the ecosystem as well. This means that among the leavers, there are contributors who are highly specialized, meaning they have large contribution to important projects of the ecosystem. The departure of such people from the ecosystem indicates potential risks for the ecosystem's evolution when a large number of important core contributors abandon the ecosystem.

III. Ecosystem migration

How does the social part of the ecosystem evolve?

We found that the number of contributors of base projects increases until February 2014, while afterwards a drop is observed. Concerning contributor activity in forks, a similar trend is observed. Also, the number of contributors abandoning the ecosystem is rapidly increasing after November 2012 and at the same time, a decrease of the number of new contributors is observed. These observations combined with the project abandonment reveal a possible correlation between developer abandonment and project abandonment. Further investigation of contributor activity while being active in Ruby and after he abandons the ecosystem showed that most contributors work in parallel to JavaScript, Shell and Python projects on GitHub. Table II summarises the results concerning the top 10 programming languages for activity of contributors for the period when they were active in Ruby (first and second column of Table II) and when they abandoned Ruby (third and fourth columns of Table II). Also, the majority of Ruby contributors that used to contribute to JavaScript in parallel to Ruby, continue their activity on JavaScript projects after abandoning the Ruby ecosystem. We consider these observations as initial evidence of a correlation between the evolution of the Ruby and JavaScript ecosystems. In future work, we aim to investigate the extent of developer migration and the characteristics of their activity in both ecosystems.

TABLE II
Contributions to other ecosystems of Ruby abandoners

Language	Active in Ruby	Language	Abandoned Ruby
JavaScript	18,038	JavaScript	13,814
Shell	10,707	Shell	8,982
Python	10,211	HTML	8,237
CSS	9,875	Python	8,131
Java	7,363	CSS	8,082
HTML	7,056	Java	5,132
C	6,406	C	4,174
PHP	5,839	Go	3,993
VimL	5,050	VimL	3,768
C++	4,649	PHP	3,517

What are the factors affecting developer and project longevity and survival in a software ecosystem?

Our previous results revealed the need to look deeper into the dynamics of developer and project evolution in software ecosystems. Our current and future research focuses on the investigation of factors that affect the longevity and survival of each actor in the ecosystem, either contributor or project. The factors behind the contributor migration both in the different projects of the ecosystem or in other ecosystems must rely on a socio-technical view of the contributor willingness and acceptance by the community so as to determine different scenarios that indicate collaboration smells that can lead to contributors' migration. More specifically, we will investigate the newcomer acceptance and support in different projects to identify the factors leading to contributor abandonment of either the project or the ecosystem. Advanced socio-technical analyses [7] are needed to accurately measure the dynamics behind evolution and migration, considering the presence of socio-technical congruence [12], [13] and socio-technical debt [14]. Furthermore, ecosystem measurements must be combined with global measurements of the development environment, e.g., technology popularity and shifting, since such tendencies can impact and disturb the ecosystem evolution. Thus, ecosystem evolution needs to be investigated with respect to both internal and external factors.

References

[1] M. Lungu, "Towards reverse engineering software ecosystems," in *International Conference on Software Maintenance (ICSM)*, 2008, pp. 428–431.

[2] K. Blincoe, F. Harrison, and D. Damian, "Ecosystems in GitHub and a method for ecosystem identification using reference coupling," in *Working Conference on Mining Software Repositories (MSR)*. IEEE Press, 2015, pp. 202–207.

[3] A. Serebrenik and T. Mens, "Challenges in software ecosystems research," in *European Conference on Software Architecture Workshops*, 2015, pp. 40:1–40:6.

[4] G. Gousios, “The GHTorrent dataset and tool suite,” in *Working Conference on Mining Software Repositories (MSR)*, 2013, pp. 233–236.

[5] G. Gousios, M.-A. Storey, and A. Bacchelli, “Work practices and challenges in pull-based development: The contributor’s perspective,” in *International Conference on Software Engineering (ICSE)*. ACM, 2016, pp. 285–296.

[6] G. Gousios, A. Zaidman, M.-A. Storey, and A. van Deursen, “Work practices and challenges in pull-based development: The integrator’s perspective,” in *International Conference on Software Engineering (ICSE)*, 2015, pp. 358–368.

[7] T. Mens, “An ecosystemic and socio-technical view on software maintenance and evolution,” in *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2016.

[8] B. Vasilescu, A. Serebrenik, and V. Filkov, “A data set for social diversity studies of GitHub teams,” in *Working Conference on Mining Software Repositories (MSR)*, 2015, pp. 514–517.

[9] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov, “Gender and tenure diversity in GitHub teams,” in *ACM Conference on Human Factors in Computing Systems (CHI)*, 2015, pp. 3789–3798.

[10] D. Posnett, R. D’Souza, P. Devanbu, and V. Filkov, “Dual ecological measures of focus in software development,” in *International Conference on Software Engineering (ICSE)*. IEEE Press, 2013, pp. 452–461.

[11] T. Mens, “Evolving software ecosystems a historical and ecological perspective,” *Dependable Software Systems Engineering*, vol. 40, p. 170, 2015.

[12] M. Cataldo, J. D. Herbsleb, and K. M. Carley, “Socio-technical congruence: A framework for assessing the impact of technical and work dependencies on software development productivity,” in *International Symposium on Empirical Software Engineering and Measurement*. ACM, 2008, pp. 2–11.

[13] M. M. M. Syeed, K. M. Hansen, I. Hammouda, and K. Manikas, “Socio-technical congruence in the Ruby ecosystem,” in *International Symposium on Open Collaboration (OpenSym)*. New York, NY, USA: ACM, 2014, pp. 2:1–2:9.

[14] D. A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, “Social debt in software engineering: insights from industry,” *Journal on Internet Services and Applications*, vol. 6, no. 1, pp. 1–17, 2015.