Towards Laws of Software Ecosystem Evolution

Category: Work in Progress

Tom Mens and Alexandre Decan
{firstname . lastname }@umons.ac.be

COMPLEXYS Research Institute
Software Engineering Lab, University of Mons

Abstract

We carry out a quantitative empirical comparison of the macro-level evolution of software packaging ecosystems for a multitude of different programming languages. We report on the most important observed differences and commonalities in the evolution of their package dependency networks. We hypothesise that the observed commonalities emerge due to the ecosystem scale and complexity. Inspired by Lehman's laws of software evolution, we seek evidence for a series of empirically observable "laws of software ecosystem evolution".

1 Lehman's Evolution Laws

The famous "laws of software evolution" [Leh78, Leh80, LPF98] have been the subject of a lot of research in the software engineering research community. It is fair to say that they have given rise to the, now well-established, subdomain of software evolution. Following the emergence of the open source phenomenon, many researchers have tried to verify whether the laws remain valid in large, distributed, open source software systems [GT00, Koc07, FRLWC08, IF10, HRRGB13]. The law of "continuing change" and the law pf "continuous growth" have shown to be valid for such systems. The law of "increasing complexity" does not appear to hold, however. It is less clear whether the other laws, such as the law of "declining quality" remain valid.

Differences in evolution dynamics have been observed depending on the considered level of granularity. At a macro-level scale, some systems (like the Linux OS) followed a superlinear growth curve [GT00], while the growth rate at the level of individual modules remained linear. Therefore, Gall et al. [GJKT97] stressed the need to study software evolution at different levels of granularity. Indeed, differences in research outcome depending on the level of detail have caused similar discrepancies in other empirical software engineering work [PFD11].

In our work, we therefore study software evolution at a macro level [GRM+09] in order to determine whether we can observe the emergence of common evolution patterns that are not directly observable at lower levels of granularity. This research goal is directly in line with the recommendation made in [HRRGB13]: *"Had Lehman started his research today, with the rich environment that we enjoy, he probably [...] would have looked for invariant properties using new statistical and empirical approaches, [...] obtaining models that would help in the development and maintenance of software."*

2 Software Packaging Ecosystems

The level of granularity we are interested in are so-called *software ecosystems*, large and coherent collections of interdependent software components that are maintained by large and geographically distributed communities of collaborating contributors. They have become a very active research domain [MS03, Lun08, MH13, BCD+13, JCB13].

Our own recent focus in this domain has been on what we refer to as *packaging ecosystems*. These are software ecosystems consisting of collections of interdependent and versioned software packages that are distributed by an automated *package manager* tool. Such ecosystems are commonplace for programming languages. Nearly every contemporary programming language has one or more central software repositories through which software developers using these language can distribute or use software developed

Proceedings of the Seminar Series on Advanced Techniques and Tools for Software Evolution SATToSE 2017 (sattose.org).
07-09 June 2017, Madrid, Spain.

for and with this language. Well-known examples are CRAN for the R programming language, npm for JavaScript and RubyGems for Ruby. We have empirically studied the structure and evolution of their package dependency networks [DMCG15, DMCG16, DMC16, DMC17] and observed some important differences in their evolution dynamics, but also some interesting commonalities.

We report on initial results of an extensive empirical comparison that we are conducting on the evolution of the package dependency networks in seven packaging ecosystems for different programming languages: Cargo for Rust, CPAN for Perl, CRAN for R, npm for JavaScript, NuGet for the .NET development platform, Packagist for PHP, and RubyGems for Ruby. Their package dependency networks were extracted based on data from the open source discovery service Libraries.io[1] under Creative Commons licence CC BY-SA 4.0[2].

The considered dependency networks vary in size and age. For example, Cargo is the smallest and youngest one, created in 2014 and containing over 9k distinct packages and over 150k dependencies in April 2017. npm is much larger, with over 462k packages and over 1,369k dependencies, while still being fairly young (created in 2010). cran is one of the oldest ecosystems, created in 1997, but still one of the smallest, with only 12k packages and 164k dependencies.

3 Preliminary Evidence of the Laws of Ecosystem Evolution

Based on a series of metrics to measure specific characteristics of the package dependency networks, we compare the evolution dynamics of these packaging ecosystems. We observe some important differences in the ecosystem dynamics, but we also observe some important commonalities, despite the different characteristics of the considered ecosystems.

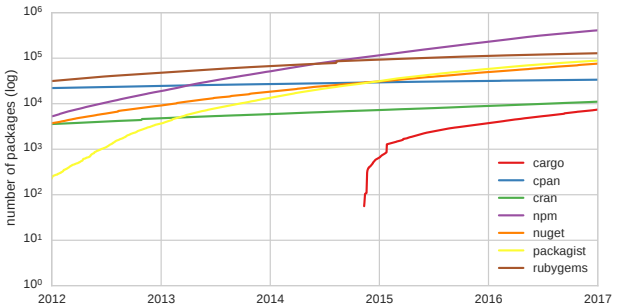
We argue that the observed commonalities are intrinsic to the scale, complexity and highly socio-technical nature of the observed ecosystems [Men12], and correspond to emerging properties that may be generally applicable to any similar software ecosystem of this size. These emerging properties, if empirically validated on a sufficiently large number of different ecosystems, have the potential of becoming established "laws of software ecosystem evolution", inspired by Lehman's laws of software evolution.

[1] http://libraries.io

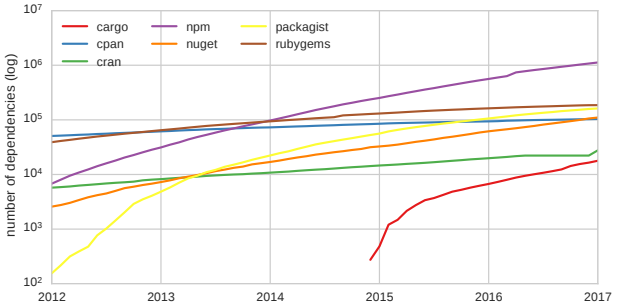
[2] https://creativecommons.org/licenses/by-sa/4.0/

3.1 Continuing Growth and Continuing Change

In Figure 1 we observe evidence of the law of *continuing growth* in terms of number of packages or number of dependencies. This law appears to be satisfied for all considered packaging ecosystems, regardless of their size.



(a) Evolution of number of packages over time



(b) Evolution of number of dependencies over time (for the latest release of each package only).

Figure 1: Continuing growth of packaging ecosystems. The y-axis uses a log-scale.

In a similar vein, Figure 2 provides evidence of the law of *continuing change*. Change is measured at ecosystem level by counting the monthly number of package updates. We observe that the number of package updates either remains stable or tends to grow over time.

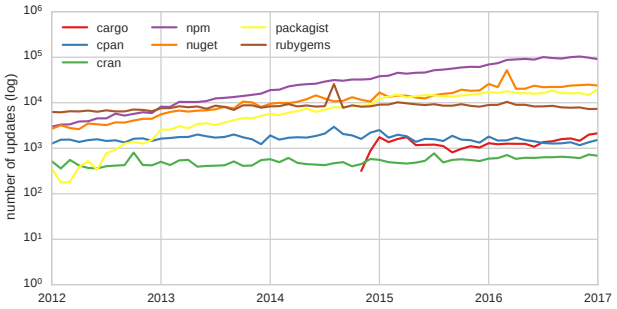


Figure 2: Evolution of number of package updates by month (using a logarithmic y-axis).

3.2 Increasing Complexity

Lehman's law of *increasing complexity* asserts that the complexity of a software system tends to increase unless if work is done (e.g. refactoring) to maintain or reduce it.

The complexity of a package dependency network is mainly determined by its dependencies. Ecosystem complexity metrics therefore need to be based on these dependencies, but they also need to be independent of the ecosystem size in order to allow for an easy comparison of the complexity between ecosystems. We propose two such simple complexity metrics in Equations 1 and 2.

$$\frac{\text{total number of direct dependencies}}{\text{total number of packages}} \quad (1)$$

$$\frac{\text{total number of transitive dependencies}}{\text{total number of direct dependencies}} \quad (2)$$

Figure 3 presents the evolution of the first metric. It provides preliminary evidence that the ecosystem complexity increases over time, in the sense that the number of direct dependencies grows faster than the number of packages.

Figure 3 shows the evolution of the second metric, and sheds more light on the "hidden complexity" induced by the presence of transitive dependencies (packages that depend on other packages that depend on other packages and so on). We observe that, according to this metric, npm, nuget and cargo can be considered more complex than the other ecosystems due to the higher ratio of transitive dependencies. Moreover, their complexity is growing faster than for the other ecosystems.

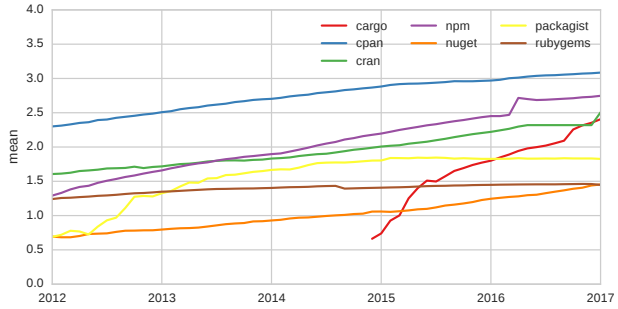


Figure 3: Evolution of the ratio of number of direct dependencies over number of packages.

It is interesting to observe that, after March 2016, ecosystem-wide effort has been undertaken by package maintainers to reduce the complexity of npm. This happened after the famous incident caused by the sudden and unexpected removal of the left-pad package [Sch16]: *"This impacted many thousands of projects. [...] We began observing hundreds of failures per minute, as dependent projects – and their dependents, and their dependents... – all failed when requesting the now-unpublished package."*

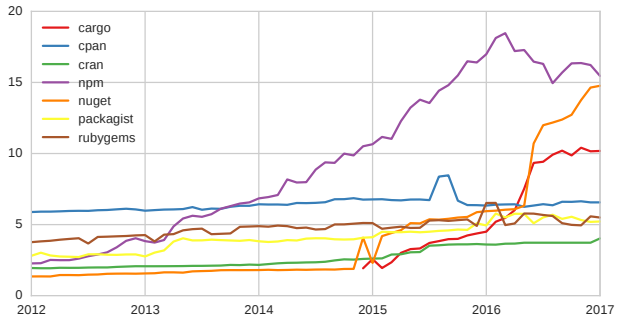


Figure 4: Evolution of the ratio of number of transitive dependencies over number of direct dependencies.

We can actually quantify the number of *high impact* packages, i.e., those packages that are most likely to affect a large proportion of the ecosystem if they become subject to an important failure or vulnerability. To do so, we define the *relative impact* of a package as the proportion of packages in the dependency network that transitively depend on it. Figure 5 shows the evolution of the number of packages p with a relative impact higher than 5% of all packages.[3] This figure resembles Figure 4 and shows that the number of high impact packages is fairly low and only very slightly increasing for most ecosystems, but higher and increasing much faster for npm, NuGet and Cargo.

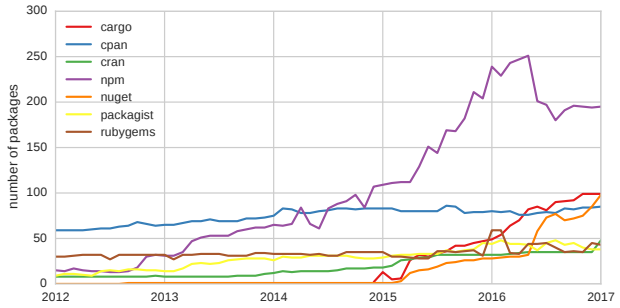


Figure 5: Evolution of the number of packages p with $RI(p) > 5\%$.

4 Conclusion

We started to explore to which extent Lehman's laws of software evolution can be generalised at the macro-level of software ecosystems. In particular, we focused on package dependency networks of large collections of software packages distributed by package manager systems for different programming languages.

[3] We obtained similar results for other percentages.

Based on historical analyses of the evolution of these dependency networks we found preliminary empirical evidence of similar laws of software ecosystem evolution. For other laws, such as the law of *declining quality*, additional sources of information are required to provide empirical evidence.

We will continue to collect more empirical evidence, and we also plan to consider the effect of the social dimension, by studying how the ecosystem contributor community influences the evolution dynamics of the package dependency networks.

Acknowledgements

This research is financed by ARC research project AUWB-12/17-UMONS-3 "Ecological Studies of Open Source Software Ecosystems" as well as FNRS Research Credit J.0023.16 "Analyis of Software Project Survival". We express our gratitude to Andrew Nesbitt and Ben Nickolls from libraries.io, who kindly granted us access to their package manager dependency data.

References

[BCD+13] Gabriele Bavota, Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella. The evolution of project inter-dependencies in a software ecosystem: the case of Apache. In *Int'l Conf. Software Maintenance*, 2013.

[DMC16] Alexandre Decan, Tom Mens, and Maelick Claes. On the topology of package dependency networks — a comparison of three programming language ecosystems. In *European Conf. Software Architecture Workshops*. ACM , 2016.

[DMC17] Alexandre Decan, Tom Mens, and Maëlick Claes. An empirical comparison of dependency issues in OSS packaging ecosystems. In *Int'l Conf. Software Analysis, Evolution, and Reengineering*, pages 2–12, 2017.

[DMCG15] Alexandre Decan, Tom Mens, Maëlick Claes, and Philippe Grosjean. On the development and distribution of R packages: An empirical analysis of the R ecosystem. In *European Conf. Software Architecture Workshops*, pages 41:1–41:6, 2015.

[DMCG16] Alexandre Decan, Tom Mens, Maelick Claes, and Philippe Grosjean. When GitHub meets CRAN: An analysis of inter-repository package dependency problems. In *Int'l Conf. Software Analysis, Evolution, and Reengineering*, pages 493–504. IEEE, March 2016.

[FRLWC08] Juan Fernández-Ramil, Angela Lozano, Michel Wermelinger, and Andrea Capiluppi. *Software Evolution*, chapter Empirical Studies of Open Source Evolution, pages 263–288. Springer, 2008.

[GJKT97] Harald Gall, Mehdi Jazayeri, René Klösch, and Georg Trausmuth. Software evolution observations based on product release history. In *Int'l Conf. Software Maintenance*, pages 160–166. IEEE, 1997.

[GRM+09] Jesús M. González-Barahona, Gregorio Robles, Martin Michlmayr, Juan José Amor, and Daniel M. Germán. Macro-level software evolution: a case study of a large software compilation. *Empirical Software Engineering*, 14(3):262–285, 2009.

[GT00] Michael W. Godfrey and Qiang Tu. Evolution in open source software: A case study. In *Int'l Conf. Software Maintenance*, pages 131–142. IEEE, 2000.

[HRRGB13] Israel Herraiz, Daniel Rodriguez, Gregorio Robles, and Jesus M. Gonzalez-Barahona. The evolution of the laws of software evolution: A discussion based on a systematic literature review. *ACM Comput. Surv.*, 46(2):28:1–28:28, 2013.

[IF10] Ayelet Israeli and Dror G. Feitelson. The Linux kernel as a case study in software evolution. *J. Systems and Software*, 83(3):485–501, 2010.

[JCB13] Slinger Jansen, Michael Cusumano, and Sjaak Brinkkemper, editors. *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*. Edward Elgar, 2013.

[Koc07] Stefan Koch. Software evolution in open source projects—a large-scale investigation. *J. Software Maintenance and Evolution*, 19(6):361–382, 2007.

[Leh78] Meir Manny Lehman. Programs, cities, students, limits to growth? *Programming Methodology*, pages 42–62, 1978. Inaugural Lecture.

[Leh80] Meir Manny Lehman. On understanding laws, evolution and conservation in the large program life cycle. *J. Systems and Software*, 1(3):213–221, 1980.

[LPF98] Meir Manny Lehman, Dewayne E. Perry, and Juan Fernandez Ramil. On evidence supporting the FEAST hypothesis and the laws of software evolution hypothesis and the laws of software evolution. In *Int'l Symp. Software Metrics*. IEEE, 1998.

[Lun08] Mircea Lungu. Towards reverse engineering software ecosystems. In *Int'l Conf. Software Maintenance*, pages 428–431, 2008.

[Men12] Tom Mens. On the complexity of software systems. *IEEE Computer*, July 2012.

[MH13] Konstantinos Manikas and Klaus Marius Hansen. Software ecosystems: A systematic literature review. *J. Systems and Software*, 86(5):1294–1306, May 2013.

[MS03] D.G. Messerschmitt and C. Szyperski. *Software ecosystem: Understanding and indispensable technology and industry*. MIT Press, 2003.

[PFD11] Daryl Posnett, Vladimir Filkov, and Premkumar Devanbu. Ecological inference in empirical software engineering. In *Int'l Conf. Automated Software Engineering*, pages 362–371. IEEE, 2011.

[Sch16] Isaac Z. Schlueter. The npm blog: kik, left-pad, and npm. `http://blog.npmjs.org/post/141577284765/kik-left-pad-and-npm`, March 2016.