ECOS: Ecological Studies of Open Source Software Ecosystems

Tom Mens, Maëlick Claes, Philippe Grosjean
COMPLEXYS Research Institute, University of Mons
7000 Mons, Belgique
firstname.lastname@umons.ac.be

***Abstract*—Software ecosystems, collections of projects developed by the same community, are among the most complex artefacts constructed by humans. Collaborative development of open source software (OSS) has witnessed an exponential increase in two decades. Our hypothesis is that software ecosystems bear many similarities with natural ecosystems. While natural ecosystems have been the subject of study for many decades, research on software ecosystems is more recent. For this reason, the ECOS research project aims to determine whether and how selected ecological models and theories from natural ecosystems can be adapted and adopted to understand and better explain how OSS projects (akin to biological species) evolve, and to determine what are the main factors that drive the success or popularity of these projects. Expressed in biological terms, we wish to use knowledge on the evolution of natural ecosystems to provide support aiming to optimize the fitness of OSS projects, and to increase the resistance and resilience of OSS ecosystems.**

I. Introduction

In this paper we present ECOS, an interdisciplinary research project involving two research teams from the COMPLEXYS Research Institute of the University of Mons. COMPLEXYS promotes interdisciplinary research collaborations on the topic of complex systems.

The ECOS project targets fundamental research on natural and software ecosystems. By relying on research in ecology, the medium-term goal is to increase the understanding of how software ecosystems evolve over time, and how software projects survive within the ecosystems of which they are part. The long-term goal is to use this insight to provide better tool support to improve upon the daily practices of software communities.

The notion of software ecosystem we adhere to in our research is the one by Messerschmitt [1] "a collection of software products that have some given degree of symbiotic relationships", which is consistent with the alternative definition by Lungu [2] "a collection of software projects which are developed and evolve together in the same environment."

II. About the Project

The five-year project is entitled *Ecological Studies of Open Source Software Ecosystem* (acronym: *ECOS*). It received a total budget of 500K euros, financed by the French community of Belgium. It started in October 2012, and will end in September 2017. The official project webpage is: http://informatique.umons.ac.be/genlog/projects/ecos.

ECOS has two financed research partners. The *Software Engineering Lab* from the Department of Computer Science, lead by Prof. Tom Mens, is specialized in software evolution research and empirical studies of open source software systems. *EcoNum*, the Lab on *Numerical Ecology of Aquatic Systems* from the Department of Biology, is specialized in biostatistics and aquatic ecology. Both partners have a shared interest in the development and use of open source software, as well as the use of statistics for research purposes.

The high-level aim of ECOS is to draw inspiration from biological evolution, ecology and natural ecosystems, in order to increase understanding of the evolution of a software ecosystem as a whole, as well as the maintainability and survivability of the projects belonging to it. These insights may ultimately lead to guidelines and evolution dashboards to help software communities improve upon their daily practices.

III. Research Questions

To achieve the project's high-level goal, two main research questions that will be addressed:

A. Which control mechanisms driving natural ecosystems can be used to explain the dynamics of software ecosystems?

Research in *ecology* studies the dynamics of the interaction between living organisms (e.g., plants, animals, microorganisms) in the context of a physical environment (including light, water, rocks, soil, minerals). The dynamics of such a natural *ecosystem* [3] are traditionally represented in a *trophic chain*, governing the interaction between *consumers* (*predators*) and their *resources* (*prey*, but also space, light, air, water and so on).

The ecosystem *dynamics* is the result of a delicate balance between consumers and resources, and can be constrained from the bottom up, from the top down, or using a combination of both [4]. Bottom-up control is limited by the resources available to primary producers. Top-down control is driven by consumption by top predators. Wasp-waist control combines both with partial effects in both directions acting simultaneously. We wish to study which of these control mechanism are prevalent in open source software ecosystems, and how they affect their evolution.

Preferably, an ecosystem should have a stable *equilibrium*, i.e., it should be *resistant* to environmental changes without too much disturbances of its biological communities. *Resilient*

978-1-4799-3752-3/14 © 2014 IEEE

CSMR-WCRE 2014, Antwerp, Belgium
Project Track

ecosystems are even able to return to an equilibrium close to the original one if they get out of balance [5]. Ecosystems with a high *biodiversity* are more likely to be resilient, since some species can compensate for others that disappear [6].

B. Which mechanisms and ideas from biological evolution can be used to explain and predict how software projects evolve?

Different biological evolution theories have been proposed. Darwinian evolution [7], the prevalent theory for explaining biological evolution, is generally considered as the major mechanism driving biological speciation (i.e., one species differentiating into two). The theory relies on the notion of competition between species: all species compete for resources in the same resource pool, and strive to survive over time by increasing their fitness fur purpose.

Other evolutionary theories have been proposed, such as *reticulate evolution* [8]. According to this theory, genetic information can be transferred between two evolutionary lineages that originated from the same ancestor species. This theory allows to explain events such as hybrid speciation (by which two lineages recombine to create a new one) and horizontal gene transfer (by which genes are transferred across species). Another evolutionary theory, originating from studies on coral reefs [9] is the *hologenome* theory, in which the object of natural selection is not the individual organism, but the organism together with its associated microbial communities.

The question is which of these theories can be adapted and adopted to best explain how projects in a software ecosystem "survive" over time. While biological evolution theories based on competition between species may perhaps be used to explain business ecosystems [10], they are not necessarily easy to transpose in an open source software setting, where the dynamics and evolution of the software ecosystem is often governed by *collaboration* between projects, rather than *competition*. The hologenome theory may perhaps be more closely related to what one observes in software ecosystems, where one should not consider the software project in isolation as the object of evolution, but rather study how it co-evolves with its associated community of contributors (e.g., users and developers). The theory of reticulate evolution may also have its merits, as it may help to explain the transfer of knowledge across projects in an ecosystem (through the sharing, migration and communication of their contributors), as well as the sharing of code (either through reuse and code cloning, or through branching and merging in version repositories).

IV. Drawing the Analogy

In [11] we have made a detailed comparison between the biological concepts of ecology, evolution and ecosystem and their software engineering counterparts. We summarize the discussion here.

A. Species = Projects

To some extent, we can consider the software projects of a software ecosystem as the counterpart of the living *species* in a natural ecosystem. (This analogy is not perfect, since a software project has no direct counterpart for the set of individuals that make up the population of a living species.)

The *environment* of the software ecosystem consists of the set of all hardware and software resources required for developing, testing, deploying and executing the software projects. The environment also includes the contributors to the software projects, as they constitute the human *resources* whose time and effort is *consumed* by the software project. A typical software ecosystem has a layered structure, similar to the *trophic chain*. At the bottom layer of the software ecosystem, we have the set of projects and libraries that constitute the core architecture on which all other projects need to depend. At the top layer we have the end-user projects on which no other project of the ecosystem depends.

Software ecosystems should be resistant to environmental disturbances that may be caused by the loss of human resources, for example because some of the core developers have lost interest in the ecosystem and prefer to spend their effort in other, competing ecosystems. Environmental changes may also be due to changes in the development process and the technology used (new programming languages, new versions of the OS, new development and version control tools). The biodiversity, allowing a software ecosystem to withstand environmental changes, could be guaranteed by having a wide diversity of different projects. This diversity could take many different forms: diversity in the functionality offered, in the programming language used, in the supported operating system, in the end-users targeted by the project, and so on.

B. Species = Contributors

If one considers a software system as a socio-technological network [12], [13], which is often the case in empirical studies on open source software, it is possible to draw another analogy, based on the duality principle [14], [15]. Rather than considering projects as species, and project contributors as the resources consumed by the projects, one could do the opposite. The species of a software ecosystem would then be the contributors (i.e., all members of the ecosystem community, including core developers, peripheral developers and end users), whereas the projects would be part of the resources that are consumed and produced by them.

Environmental disturbances may cause a loss of resources (e.g., legacy projects may no longer be viable because the operating system for which they were developed has been replaced). The ecosystem will be more resilient if its community is sufficiently diverse (e.g., containing developers that are specialized in different programming languages, operating systems; but also containing contributors specialized in other activities such as testing, debugging, documentation, translation, communication, and so on).

In this analogy, the trophic web would look a bit like the well-known onion model, containing a small number of core developers at the bottom of the food chain, many more peripheral developers higher up in the chain, until we reach the end-users that only consume but do not produce anything.

Some software ecosystems may be bottom-up controlled, primarily driven by input from its core developers, or have limited budget, hardware resources and human resources. Other ecosystems may be mainly top-down controlled, with most changes driven by change requests and bug reports coming from the end users. A study and understanding of how the type of control affects the software ecosystem dynamics may lead to better management strategies of OSS projects.

V. Work in Progress

A. The Gnome Ecosystem

The research carried out in the context of this project is a natural continuation of our former research on the evolution of open source software ecosystems, mainly using the Gnome software ecosystem as a case study [16], [17]. Gnome was selected because of its longevity (over 15 years of historical data available), and size (over 1300 Gnome projects and several thousands of contributors). Directly related to the notion of biodiversity, we studied how Gnome contributors specialize themselves in different activities [14]. We identified two important subgroups of Gnome contributors: the *coders* that limit their attention to a small number of projects on which they are very active, and the *translators* that take care of localization and internationalization and do this for a large number of different projects.

To gain understanding of the resilience of the Gnome ecosystem, we are studying the migration patterns of Gnome contributors, in collaboration with Uzma Raja from the University of Alabama [18]. A sudden loss (or intake) of contributors observed in the Gnome ecosystem may be indicative of an important environmental disturbance, so we can analyse if and how the ecosystem returned to an equilibrium after this disturbance. We can also study the migration patterns of contributors between Gnome projects, as these are likely to play and important role in the survivability of projects within the Gnome ecosystem. The graph in Fig. 1 visualizes some of these migrations at a given snapshot, where each node represents a Gnome project, and each edge represents the migration of contributors to another project. The weight of the edge is proportional to the number of migrations.

B. The CRAN Ecosystem

We are also studying the ecosystem of R packages, contained in CRAN (cran.r-project.org), the Comprehensive R Archive Network. To this extent, we are collaborating with the research teams of Daniel German and Bram Adams in Canada, who have studied this ecosystem in the recent past [19]. Like Gnome, CRAN has a history of over 15 years, and it contains more than 5000 packages, maintained by a community of over 2500 contributors. A particularity of this ecosystem is that the community is mainly composed of end-users, that do not necessarily have a lot of development experience. CRAN has been experiencing maintainability problems due to the superlinearly growing number of packages [20] and the limitations of R's dependency versioning system [21].

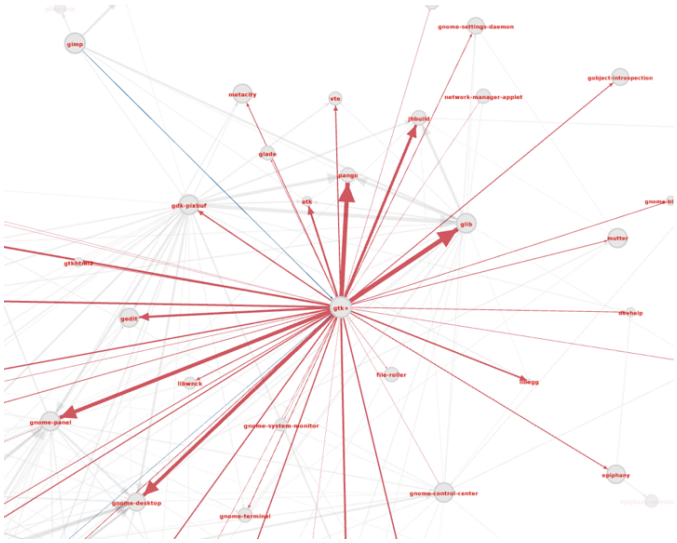

Fig. 1. Migration of contributors across Gnome packages

To give an indication of the complexity of the CRAN ecosystem, Fig. 2 visualizes all contributed packages in CRAN and their required dependencies to other contributed packages. Dependencies to the base packages are not shown, in order not to clutter the image. We clearly observe a "halo" of packages that do not depend on any other contributed package.

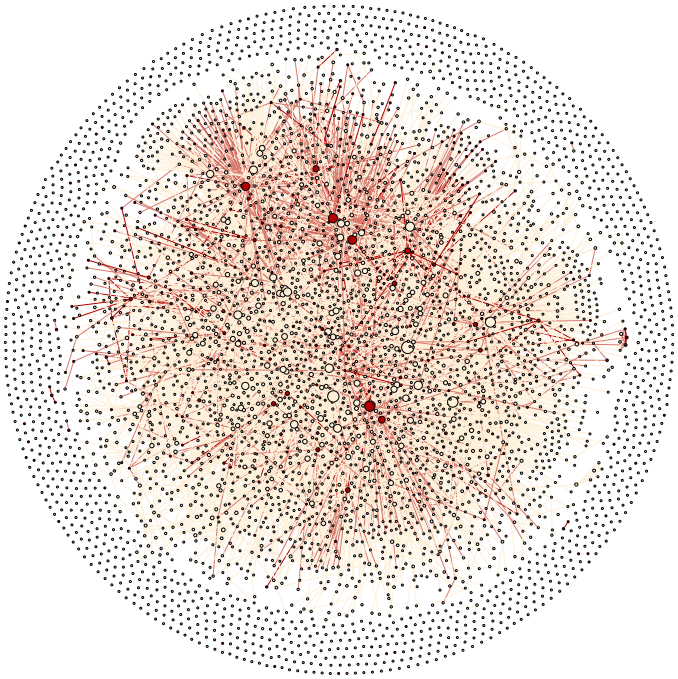

Fig. 2. Dependencies between CRAN packages. Red arrows indicate dependencies to packages that have an error status.

For the CRAN ecosystem, we are currently studying the factors that affect the survivability of packages. Two particular factors of interest are the reliability (how frequently does the package have errors) and maintainability (how long does it

take to fix an error). Some preliminary research results have been reported in [22].

VI. Related Work

Various researchers have started to use measures and techniques from ecology in order to get a better understanding of software development and software evolution. We only discuss related work here that directly relies on such ideas. Probably the closest related to our project is the European FP7 FET *DIVERSIFY* research project (diversify-project.eu, 2013-2016) exploiting the analogy with ecological systems, biodiversity and evolutionary ecology in order to come to better design principles for software-intensive distributed systems. The objective is to increase diversity in software systems for increased resilience and the ability to react to unpredicted events [23], [24].

Hutchins [25] used genetic algorithms to understand evolutionary software development processes. Each branch of a software project is compared to an individual of a biological species and merging of branches is similar to the crossover operation (reproduction of two individuals). Raja [18] proposed and validated a new multi-dimensional measure of OSS project survivability, called *project viability*, in terms of *vigor* (the ability of a project to evolve over a period of time), *resilience* (the ability of a project to respond to perturbations), and *organization* (the amount of structure in a project). Calzolari et al. [26] empirically studied and predicted the evolution of software defects in two industrial software systems by means of the dynamic predator-prey model. Software defects (requiring corrective actions) can be seen as the equivalent of biological prey, whereas the programmers (consuming the defects by correcting them) act as predators. Some adaptations to the predator-model were needed since, unlike species, software defects cannot reproduce themselves. Similarly, Posnett et al. [15] compared a developer-module contribution network to a predator-prey food web, by considering software modules as predators that feed upon the limited cognitive resources of developers (their prey). They drew upon ideas from biodiversity to measure how focused the activities on a module are, as well as how focused the activities of a developer are. More specifically, they defined metrics based on the notions of *specialization*, *diversity* and relative entropy used in ecology [27]. They found empirical evidence that more focused developers introduce fewer defects. Conversely, increased module activity focus leads to a larger number of defects.

Acknowledgment

This work is partially supported by research project AUWB-12/17-UMONS-3, an Action de Recherche Concertée financed by the Ministère de la Communauté française - Direction générale de l'Enseignement non obligatoire et de la Recherche scientifique, Belgium.

References

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

[2] M. Lungu, "Towards reverse engineering software ecosystems," in *Int'l Conf. Software Maintenance*, 2008, pp. 428–431.

[3] A. J. Willis, "The ecosystem: an evolving concept viewed historically," *Functional Ecology*, vol. 11, pp. 268–271, 1997.

[4] G. Hunt and S. McKinnell, "Interplay between top-down, bottom-up, and wasp-waist control in marine ecosystems," in *Progress In Oceanography*, vol. 68, no. 2-4, 2006, pp. 115–124.

[5] C. S. Holling, "Resilience and stability of ecological systems," *Annual Review of Ecology and Systematics*, vol. 4, pp. 1–23, 1973.

[6] K. S. McCann, "The diversity-stability debate," *Nature*, vol. 405, pp. 228–233, 2000.

[7] C. Darwin, *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, Nov. 1859.

[8] C. R. Linder, B. M. E. Moret, L. Nakhleh, and T. Warnow, "Network (reticulate) evolution: biology, models, and algorithms," in *Pacific Symp. Biocomputing*, 2004.

[9] E. Rosenberg, O. Koren, L. Reshef, R. Efrony, and I. Zilber-Rosenberg, "The role of microorganisms in coral health, disease and evolution," *Nature Reviews Microbiology*, vol. 5, no. 5, pp. 355–362, 2007.

[10] S. Jansen, M. Cusumano, and S. Brinkkemper, Eds., *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*. Edward Elgar, 2013.

[11] T. Mens, M. Claes, P. Grosjean, and A. Serebrenik, *Evolving Software Systems*. Springer, 2014, ch. Studying Evolving Software Ecosystems based on Ecological Models.

[12] A. Sarma, L. Maccherone, P. Wagstrom, and J. Herbsleb, "Tesseract: Interactive visual exploration of socio-technical relationships in software development," in *Int'l Conf. Software Engineering*, 2009, pp. 23–33.

[13] D. Surian, Y. Tian, D. Lo, H. Cheng, and E.-P. Lim, "Predicting project outcome leveraging socio-technical network patterns," in *European Conf. Software Maintenance and Reengineering*, 2013.

[14] B. Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens, "On the variation and specialisation of workload: A case study of the Gnome ecosystem community," *J. Empirical Software Engineering*, pp. 1–54, 2013.

[15] D. Posnett, R. D'Souza, P. Devanbu, and V. Filkov, "Dual ecological measures of focus in software development," in *Int'l Conf. Software Engineering*. IEEE, 2013, pp. 452–461.

[16] M. Goeminne and T. Mens, *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*. Edward Elgar, 2013, ch. Analyzing ecosystems for open source software developer communities.

[17] M. Goeminne, M. Claes, and T. Mens, "A historical dataset for the GNOME ecosystem," in *Int'l Conf. Mining Software Repositories*. IEEE Computer Society, 2013, pp. 167–170, https://bitbucket.org/mgoeminne/sgl-flossmetric-dbmerge.

[18] U. Raja and M. J. Tretter, "Defining and evaluating a measure of open source project survivability," *IEEE Trans. Soft. Eng.*, vol. 38, no. 1, pp. 163–174, 2012.

[19] D. M. Germán, B. Adams, and A. E. Hassan, "The evolution of the R software ecosystem," in *European Conf. Software Maintenance and Reengineering*, 2013, pp. 243–252.

[20] K. Hornik, "Are there too many R packages?" *Austrian Journal of Statistics*, vol. 41, no. 1, pp. 59–66, 2012.

[21] J. Ooms, "Possible directions for improving dependency versioning in r," *R Journal*, vol. 5, no. 1, pp. 197–206, Jun. 2013.

[22] M. Claes, T. Mens, and P. Grosjean, "On the maintainability of CRAN packages," in *European Conf. Software Maintenance and Reengineering*. IEEE Computer Society, 2013.

[23] B. Baudry and M. Monperrus, "Towards ecology inspired software engineering," INRIA, Tech. Rep. 7952, May 2012.

[24] D. Mendez, B. Baudry, and M. Monperrus, "Empirical evidence of large-scale diversity in api usage of object-oriented software," in *Working Conf. Source Code Analysis and Manipulation*. IEEE Computer Society, 2013.

[25] D. Hutchins, "A biologist's view of software evolution," in *Reflection, AOP and Meta-Data for Software Evolution*, 2005, pp. 95–105.

[26] F. Calzolari, P. Tonella, and G. Antoniol, "Maintenance and testing effort modeled by linear and nonlinear dynamic systems," *Information and Software Technology*, vol. 43, no. 8, pp. 477 – 486, 2001.

[27] N. B. Nico Blüthgen, Florian Menzel, "Measuring specialization in species interaction networks," *BMC ecology*, vol. 6, no. 1, p. 9, 2006.