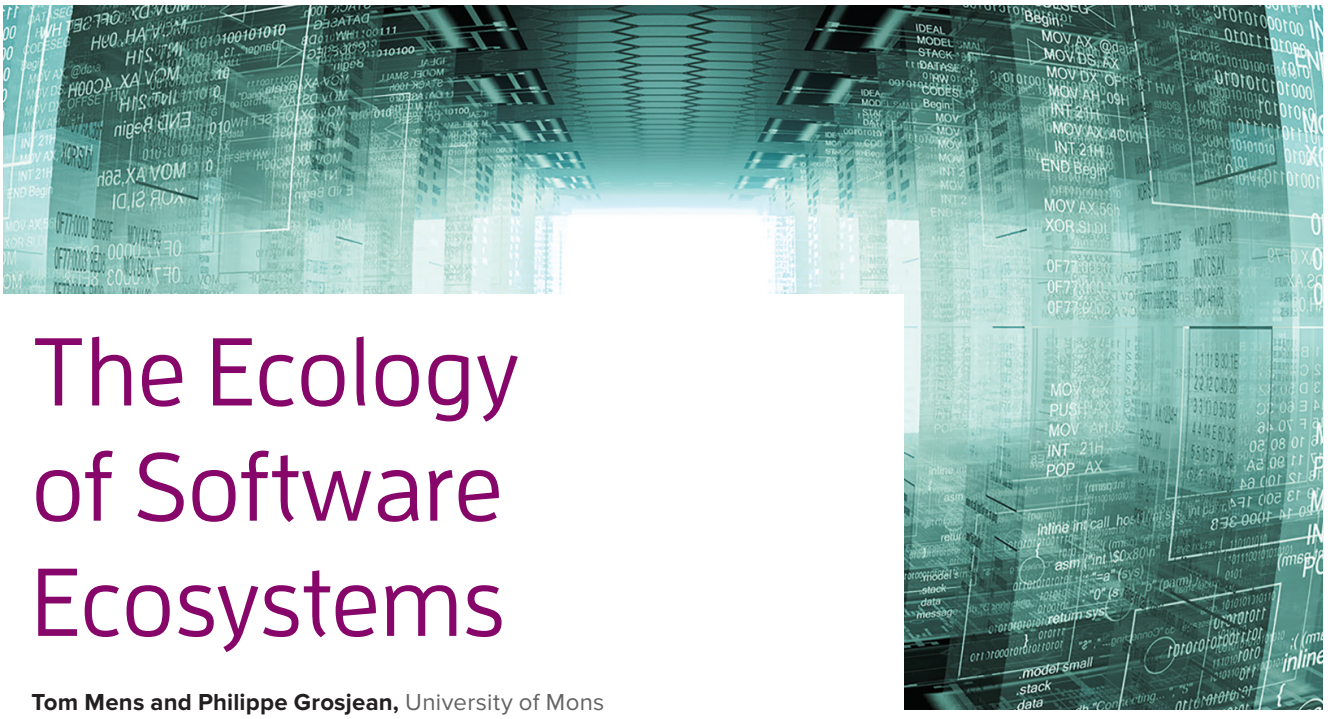
EDITOR **MIKE HINCHEY**
Lero—the Irish Software Research Centre;
mike.hinchey@lero.ie



The Ecology of Software Ecosystems

Tom Mens and Philippe Grosjean, University of Mons

Software ecosystems—collections of software projects developed and used by the same community—are extremely complex. Comparing them with biological ecosystems can yield new strategies for improving their effectiveness and resilience.

The discipline of ecology studies the interactions among living things in the context of their physical environment. The dynamics of these interactions are influenced by energy, nutrients, gas exchange, and temperature, as well as other organic and inorganic materials in the environment. For example, a wide variety of marine species in coral reefs are supported in a fragile environmental equilibrium; even small temperature fluctuations can have mortal consequences within such an ecosystem.

Biological ecosystem dynamics are traditionally represented by a trophic web, reflecting the energy flow among producers, consumers, and the environment; shifts in species populations and available resources affect an ecosystem's delicate equilibrium (see Figure 1a). An ecosystem should be robust enough to maintain a stable equilibrium despite fluctuating dynamics to support its biological communities.

SOFTWARE ECOSYSTEM

In their book *Software Ecosystem*, David Messerschmitt and Clemens Szyperski define a software ecosystem as "a collection of software products that have some given degree of symbiotic relationships."[1] Mircea Lungu describes it as "a collection of software projects which are developed and evolve together in the same environment."[2] Well-known examples of software ecosystems include programming language archive networks, mobile app stores, and various distributions of the Linux operating system.

Given the sociotechnical nature of software development projects that involve software components and the

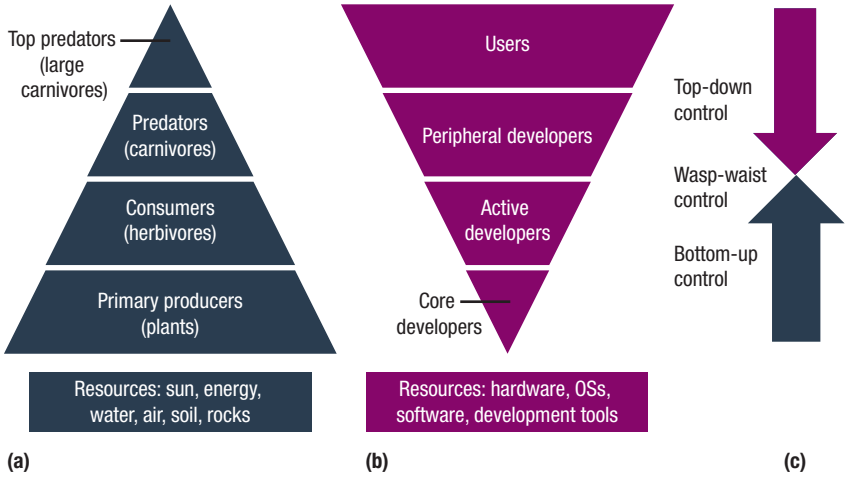


Figure 1. Schematic comparison of ecosystems. (a) Trophic web of a biological ecosystem. (b) A software ecosystem seen from a social viewpoint. (c) Ecosystems can be controlled from the bottom up—constrained by resources available to primary producers—or from the top down, driven by predators' consumption. In the wasp-waist control method, partial effects occur simultaneously from both directions.

project's contributors, including developers, testers, debuggers, and end users, software ecosystems can be compared to biological ecosystems in two ways:

- *Biological species ≈ software components*. From a technical viewpoint, software components can be compared to living species in a biological ecosystem. The software ecosystem contains all the hardware and software components required for developing, testing, deploying, and executing the software. As in a trophic web, some of these components (such as shared development platforms and software libraries) act as producers that are consumed by other software components.
- *Biological species ≈ project contributors*. In this social view, software project contributors are analogous to the living species in a biological ecosystem. As shown in Figure 1b, the project contributors form a trophic web consisting of core developers that produce the core architecture consumed by active developers, and so on. All these contributors share a common pool of project resources—such as version commits, issue reports, change requests, bug fixes, documentation, and tests.

CONTROL MECHANISMS GOVERNING ECOSYSTEM DYNAMICS

As Figure 1c illustrates, ecosystem dynamics can be controlled from the bottom up, constrained by the resources available to primary producers. Alternatively, they can be constrained from the top down, driven by predators' consumption. The wasp-waist control method combines both, with partial effects occurring simultaneously from both directions.

In the technical view of a software ecosystem, the bottom of the trophic web contains the set of components (including shared libraries) that constitute the core software architecture, upon which all other software components depend. In the social view, the trophic web contains a small number of core developers at the bottom, active developers higher up, peripheral developers even higher, and end users at the top, whose consumption creates the demand for more developers.

Some software ecosystems are primarily driven by input from core developers, with constraints related to limited resources such as budget, hardware, and personnel, or bottom-up controlled. Other software ecosystems are driven largely by change requests and bug reports coming from end users, or top-down controlled. Empirical studies aimed at understanding the relationship between control mechanisms and software ecosystem dynamics will lead to better software project management strategies.

ECOSYSTEM DIVERSITY

Resilience describes an ecosystem's ability to return to equilibrium once it's knocked out of balance. A biological ecosystem with high diversity is more likely to be resilient because some species compensate for others that disappear. Biodiversity could also play an important role in the success and resilience of software ecosystems, from both the technical and social points of view.

In the technical analogy, environmental perturbations can be caused by the loss of resources, either due to lack of interest or competing software components that become higher priority. They can also be caused by changes in the development process and the technology used, such as the introduction of new programming languages, new OS versions, and new development and version control tools. Accommodating a wide variety of software components could increase the software ecosystem's resilience or ability to adapt to change. This diversity could take many forms: the functionality offered, the programming language used, the supported OS, the end users targeted by the software, and so on.

In the social view, environmental disturbances can also lead to a loss of resources—for example, legacy projects might no longer be viable because the OS or programming language for which they were developed has become outdated. Increased diversity among the contributing developers—such as fluency in different programming languages and OSs or skill sets that include a range of activities including testing, debugging, documentation, translation, and so on—improves the ecosystem's resilience.

Many different metrics have been proposed to measure species diversity. Daryl Posnett and his colleagues relied on measures of relative entropy to study the activity focus of each software component (and the activity focus of each project contributor), and related this to the likelihood of introducing software defects into these components.[3] In a recent survey, Benoit Baudry and Martin Monperrus explored the different facets of software diversity, stressing the importance of identifying the underlying principles driving diversity's constant renewal.[4]

SURVIVAL OVER TIME

The theory of Darwinian evolution is generally used to describe the major mechanism driving *biological speciation*, which is when one species differentiates into two. According to this theory, all species compete in the same resource pool and survive by increasing their fitness or ability to reproduce. Over time, the most competitive genetic traits proliferate. This is akin to the evolution of software ecosystems, in which successful traits continue to proliferate and less successful traits disappear from future software versions.

In statistical ecology, the technique of survival analysis has been used to estimate, compare, and model the survival rates of animal populations in their ecosystems. Survival analysis is also gaining traction in software ecosystem research, determining the main factors that predict survival of software components or project contributors within their ecosystem.

Although research on software ecosystems is thriving, the complex dynamics of these ecosystems are still not well understood. Adapting biological theories and measures to the context of software ecosystems will help researchers come up with new strategies to improve the effectiveness and resilience of software ecosystems.

FURTHER READING

For additional information on business software ecosystems and the usefulness of the evolution-based changes in software ecosystems, the authors recommend *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry* (Edward Elgar Publishing, 2013) and *Evolving Software Systems* (Springer, 2014).

ACKNOWLEDGMENTS

This research was conducted as part of the Belgian ARC research project AUWB-12/17-UMONS-3, entitled "Ecological Studies of Open Source Software Ecosystems."

REFERENCES

1. D.G. Messerschmitt and C. Szyperski, *Software Ecosystem: Understanding an Indispensable Technology and Industry*, MIT Press, 2003.
2. M. Lungu, "Towards Reverse Engineering Software Ecosystems," *Proc. Int'l Conf. Software Maintenance* (ICSM 08), 2008, pp. 428–431.
3. D. Posnett et al., "Dual Ecological Measures of Focus in Software Development," *Proc. IEEE Int'l Conf. Software Eng.* (ICSE 03), 2003, pp. 452–461.
4. B. Baudry and M. Monperrus, "The Multiple Facets of Software Diversity: Recent Developments in Year 2000 and Beyond," to be published in *ACM Computing Surveys*, 2015.

TOM MENS is a professor at the COMPLEXYS Research Institute at the University of Mons. Contact him at tom.mens@umons.ac.be.

PHILIPPE GROSJEAN is a professor at the COMPLEXYS Research Institute at the University of Mons. Contact him at philippe.grosjean@umons.ac.be.