Software Ecosystem Evolution: It's complex!

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

1. INTRODUCTION

This extended abstract proposes a vision statement that advocates the use of results and mechanisms from *complex systems theory* to understand and control the evolution dynamics of software ecosystems (SECOs). We argue why such an interdisciplinary take on software ecosystem evolution may ultimately lead to better techniques and tools for managing and controlling SECOs. The driving forces of the ecosystem dynamics will be both social and technical. The need to adopt such a socio-technical view is confirmed by [2], and we explored this view in more detail in [13].

Important challenges in SECO research have been identified in [2, 19]. A first one is the need to understand *how and why ecosystems emerge*. This challenge also involves the need for techniques to identify SECOs and their boundaries. Bogart et al. observed that the practices, policies and tools used by the ecosystem's actors differ significantly across SECOs [3]. A question that naturally emerges is therefore how the social and technical aspects of a SECO affect its evolution over time. It is yet unclear why some SECOs continue to survive, while others are abandoned. Understanding the dynamics of a SECO, and its *resilience* to external perturbations becomes key. By resilience we refer to the capacity of an ecosystem to absorb and recover from environmental changes and disturbances. We hypothesise that a complex systems perspective on SECOs allows to gain a better understanding in each of the above challenges, ultimately leading to better tools for supporting the ecosystem's actors.

2. COMPLEX SYSTEMS THEORY

As Melanie Mitchell beautifully illustrates in her book, complex systems abound in all domains of science (ecology, economy, biology, sociology, computer science, physics, linguistics, and many more) [14]. There is no single encompassing definition of what constitutes a *complex system*. However, the majority of complex systems have in common that they are composed of many interdependent parts that interact with each other and with their environment. By doing so, collective behaviours *emerge* that cannot be understood by studying the components in isolation. Much of the observed complexity arises from interaction dynamics.

One of the goals of *complex systems theory* is to propose and validate theoretical models that capture the essential aspects of real-life complex systems. Such models aim to explain how complex systems emerge, to understand which structural or behavioural properties they exhibit, and to predict how such systems evolve over time.

Network science or *network theory* is a subdomain of complex systems theory, in which network or graph structures are used as simplified mathematical models representing the structural or topological aspects of a complex system. Depending on the type of complex system being modelled, the nodes and edges of the network will represent different things. In a social network, for example, nodes will represent social actors (typically persons, groups or organisations) while the edges will represent interactions between these actors (e.g., communication about a specific topic, or collaboration on a joint activity or project). The field of *social network analysis* intensively studies such networked social structures [18].

Many complex networks have tend to have similar characteristics. **Skewed distributions:** The distribution of many structural network metrics (such as the in- and out-degree of nodes) tends to be highly skewed, often corresponding to some kind of *power law* or *log normal* distribution. **Scale-free topology:** The observed distributions are invariant to scale. When zooming in on part of the network, a similar structure is observed. **Small world assumption:** The average path length between any two nodes of the network is very small. **Presence of hubs and clusters:** A complex network tends to have many highly clustered components, with few hub nodes that interlink these components.

These characteristics make complex networks considerably more resilient to changes than random networks. If random nodes are removed from the network, it is very likely that these nodes will have few links (because of the power law behaviour). Because of this their removal will not impact the overall network structure. On the downside, the removal of hub nodes (those that have many links) can have quite dramatic consequences for the ecosystem.

Many different models have been proposed to explain how such scale-free complex networks emerge. One of the more popular ones is the theory of *preferential attachment* [1]. This theory is based on the assumption that networks grow in such a way that nodes with higher degree receive more

new links than nodes with lower degree.

3. COMPLEX SOFTWARE SYSTEMS

Many researchers have used complex systems theory to study the network characteristics of software systems. Valverde et al. [20], Myers et al. [15], Potanin et al. [17], Concas et al. [7] and Louridas et al. [10] found evidence of scale-free topologies and highly skewed distributions in dependency graphs of object-oriented software systems. Such properties appear to *emerge* as a side effect of the development process.

Several plausible explanations have been suggested about why software dependency networks have a scale-free structure, mostly without any strong compelling evidence: Valverde et al. [20] suggest that the emergence of scaling arises from logical optimisation process. Myers et al. [15] proposed the process of refactoring to improve the structure of existing code as a possible explanation for the emergence of scale-free networks in software. Inspired by Darwin's ideas of evolutionary adaptation, Venkatasubramanian et al. proposed a generic model based on network parameters such as efficiency, robustness, cost, and environmental selection pressure [21]. Using a genetic algorithm their model was able to generate different types of network structures, depending on the chosen parameters. Li et al. [8] proposed an extended model of preferential attachment adapted to software systems, and used it to simulate growth models that mimic the well-known design principle of low coupling and high cohesion. If software developers strive towards this principle, they will naturally obtain systems containing highly cohesive components that are lowly coupled between them, reminiscent of the hubs and clusters structure presented in Section 2.

The complex network properties of *social networks* have also been studied by many researchers. For example, López-Fernándes et al. studied the structure of the collaboration network of large open source software projects [9]. Pinzger et al. [16] and many follow-up papers explored the socio-technical network composed of software developers and the software modules they contributed to. The use of network centrality measures of this socio-technical network allowed them to come up with better defect prediction models.

4. COMPLEX SOFTWARE ECOSYSTEMS

We believe that complex systems theory can be very helpful in understanding the evolution of SECOs. Their *technical* network structure would be composed of nodes that represent the software components belonging to the SECO, while the edges represent component dependencies. The SECOs *social* network structure would be composed of nodes that present the actors of the SECO, while the edges represent interactions or collaborations between these actors. Both network structures may have complex network characteristics, and these characteristics may be different from what one observes at the level of the individual software components, packages or projects that make up the SECO, and that can be considered as complex systems themselves.

The multi-level model proposed by Li et al. for simulating software network evolution is a good starting point for further research [8]. They explain that a software system is multi-level by nature. At the lowest level, one can find for example classes and their interrelationships. At the second level, one can find mechanisms like design patterns, libraries and frameworks. At the third level there are packages, subsystems and components. Further levels can be added, if needed. Such a multi-level model naturally represents the hierarchical structure of software systems and software ecosystems, and simulations based on this model (using historical data of seem to generate more realistic simulations of software network growth. While being a good starting point, the model needs to be extended to take into account the social network as well, so that it also reflects the dynamics and growth of the ecosystem's contributor community, as well as the collaborative development process.

While the above model can be used to explain how software ecosystems continue to grow under normal circumstances, it does not provide any help if dramatic changes in the technical or social network arise. Therefore, additional models are needed to study the *resilience* of software ecosystems. To analyse such resilience, dedicated tools (e.g. web-based dashboards) are needed to analyse or predict the impact of important perturbations. For example, one could carry out *what if* analyses on the effect of removing hubs that correspond to key contributors in the social network, or to core components in the technical network of the SECO. There is a clear need for such analyses. Indeed, several examples of major disruptions in a SECO due to unexpected removal of key nodes have been reported. For example, Zanetti et al. [22] observed the sudden departure of a core contributor to Gentoo's bug tracking system, causing a major disruption in the community's bug handling performance. As another example, JavaScript's *npm* package management system was severely affected by the sudden removal of a single package (*leftpad*) on which thousands of other packages depended. Analysing the SECO network structure would allow to identify hub nodes more easily, allowing the SECO managers to plan ahead by taking appropriate actions to reduce the risk of sudden removal of such nodes, as well as to reduce the impact of such removal if it does occur.

Since different SECOs use a wide variety of different practices, policies and tools [3], it is reasonable to expect differences in the structure of their social and technical network. More specifically, network measures may reveal important differences in the characteristics that are typical of complex systems. Such observed differences in network topology may be indicative of the ecosystem's resilience. This would provide more insights in how the organisational and architectural choices made by a SECO community affect the ecosystem dynamics (e.g., its growth, ease of maintenance and survival chances). Perhaps differences in the complex network characteristics of a SECO also depend on the ecosystem's domain (e.g., programming languages). We therefore suggest to study how and why SECOs differ from a complex system point of view, and to leverage these findings for suggesting improvements in the architectural, tooling-related, and organisational choices made by a SECO.

5. SUMMARY

To summarise, we believe that research on software ecosystem evolution can benefit a lot from taking a complex systems perspective,. We propose to consider open source SECOs as complex systems, and to use measures and models from network science to study the emergent characteristics of the SECOs' socio-technical networks. We suggest to use growth models inspired by the theory of preferential attach-

ment to explain the emergent dynamics of the observed network structure. In addition, we propose to rely on models of resilience to assess and reduce the impact of major disruptions in the ecosystem. Finally, we suggest to compare the complex networks of multiple SECOs, in order to assess the impact of the architectural and organisational choices of the community on the resilience and maintainability of their SECO.

6. REFERENCES

[1] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[2] K. Blincoe, F. Harrison, and D. Damian. Ecosystems in GitHub and a method for ecosystem identification using reference coupling. In *Int'l Conf. Mining Software Repositories*, 2015.

[3] C. Bogart, C. Kästner, J. Herbsleb, and F. Thung. How to break an API: Cost negotiation and community values in three software ecosystems. In *Int'l Symp. Foundations of Software Engineering*, 2016.

[4] 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 *Int'l Symp. Empirical Software Engineering and Measurement*, pages 2–11. ACM , 2008.

[5] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb. Software dependencies, work dependencies, and their impact on failures. *IEEE Transactions on Software Engineering*, 35(6):864–878, Nov 2009.

[6] L. J. Colfer and C. Y. Baldwin. The mirroring hypothesis: Theory, evidence and exceptions. Technical Report Finance Working Paper No. 16-124, Harvard Business School, May 2016.

[7] G. Concas, M. Marchesi, S. Pinna, and N. Serra. Power-laws in a large object-oriented software system. *IEEE Trans. Soft. Eng.*, 33(10):687–708, 2007.

[8] H. Li, L.-Y. Hao, and R. Chen. Multi-level formation of complex software systems. *Entropy*, 18(178), 2016.

[9] L. Lopez-Fernandez, G. Robles, J. Gonzalez-Barahona, and I. Herraiz. Applying social network analysis techniques to community-driven libre software projects. In *Integrated Approaches in Information Technology and Web Engineering: Advancing Organizational Knowledge Sharing*, chapter 3, pages 28–50. IGI Global, 2009.

[10] P. Louridas, D. Spinellis, and V. Vlachos. Power laws in software. *ACM Trans. Software Engineering and Methodology*, 18(1):1–26, Oct. 2008.

[11] A. MacCormack, C. Baldwin, and J. Rusnak. Exploring the duality between product and organizational architectures: A test of the "mirroring" hypothesis. *Research Policy*, 41(8):1309 – 1324, 2012.

[12] K. Manikas and K. M. Hansen. Software ecosystems: A systematic literature review. *J. Systems and Software*, 86(5):1294–1306, May 2013.

[13] T. Mens. An ecosystemic and socio-technical view on software maintenance and evolution. In *Int'l Conf. Software Maintenance and Evolution*, 2016.

[14] M. Mitchell. *Complexity: A Guided Tour*. Oxford University Press, 2011.

[15] C. R. Myers. Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. *Physical Review E*, 68(4), 2003.

[16] M. Pinzger, N. Nagappan, and B. Murphy. Can developer-module networks predict failures? In *Int'l Symp. Foundations of Software Engineering*, pages 2–12. ACM, 2008.

[17] A. Potanin, J. Noble, M. Frean, and R. Biddle. Scale-free geometry in oo programs. *Commun. ACM*, 48(5):99–103, May 2005.

[18] J. Scott. *Social Network Analysis*. SAGE, 2012.

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

[20] S. Valverde, R. Ferrer Cancho, and R. V. Solé. Scale-free networks from optimal design. *Europhysics Letters*, 60, 2002.

[21] V. Venkatasubramanian, S. Katare, P. R. Patkar, and F.-P. Mu. Spontaneous emergence of complex optimal networks through evolutionary adaptation. *Computers and Chemical Engineering*, 28(9):1789–1798, 2004.

[22] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer. The rise and fall of a central contributor: Dynamics of social organization and performance in the Gentoo community. In *Int'l Workshop on Cooperative and Human Aspects of Software Engineering*, pages 49–56, May 2013.