Towards a Typification of Software Ecosystems

Jens Knodel[1(✉)] and Konstantinos Manikas[2]

[1] Fraunhofer Institute for Experimental Software Engineering (IESE),
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
jens.knodel@iese.fraunhofer.de
[2] Department of Computer Science (DIKU),
University of Copenhagen, Copenhagen, Denmark
kmanikas@di.ku.dk

Abstract. Traditionally, software engineering has been dominated by stand-alone development organizations and collaborations between contractors, integrators and suppliers. In the last decade, the notion of software ecosystems has been established as a new paradigm in software engineering. In its essence it proposes participative engineering across independent development organizations centered on a common technology.

This paper reviews the current state-of-the-art and presents a first step towards a typification of successful software ecosystems. We discuss key characteristic of the ecosystem types and present a set of example cases. The characterization reviews and consolidates existing research and discusses variations within the key building block of a software ecosystem. It further enables sharpening the borders of what an ecosystem is (and what not) and how the individual types can be differentiated. Thus, this paper contributes to widening the understanding of software ecosystems and serves to prepare a software ecosystem taxonomy.

Keywords: Software ecosystems · Software engineering · Ecosystem types · Ecosystem taxonomy

1 Introduction

Software systems have been traditionally developed by a single organization in isolation and within a collaboration of several organizations, whereby one organization subcontracted other suppliers to deliver parts of or whole software systems according to some kind of specification. Today, we can observe an increasing number of software systems that strongly gain value by contributions added by other organizations - without being bound to contracted specification of what to deliver at what point in time. Prominent examples of such so-called software ecosystems (SECOs) are for instance, Eclipse an open platform with plugins for all kinds of purposes or mobile device platforms like iOS or Android which are enriched by millions of apps.

The term software ecosystem was first coined more than a decade ago [1]. The research community has been successful in scattering various definitions

J.M. Fernandes et al. (Eds.): ICSOB 2015, LNBIP 210, pp. 60–65, 2015.
DOI: 10.1007/978-3-319-19593-3_5

of software ecosystems[1] since then. Those partially overlapping definitions define the space and the borders of the current shared understanding of software ecosystems in the research community.

In this paper, we argue for a having wider understanding on the range of existing kinds of software ecosystems. We derive our observations from ecosystems in operation (either by analysis of open, active ecosystems or of closed ecosystem where we had insights due to collaborations with industrial partners). We distill the key building blocks of software ecosystems observed and provide a first set of ecosystem types. By this we aim at paving the way towards an ecosystem taxonomy in order to enable a better understanding of ecosystems in general and its research challenges and implication in particular.

2 Setting the Scene

The work by Manikas and Hansen [7] analyzed the definitions in the literature (published until 2012). They propose a definition of software ecosystems by analyzing the existing definitions and identify three main elements that form software ecosystems: (i) common software and (a) technological platform(s), (ii) business or interests, and (iii) connecting relationships or interaction. However, today there exists a number of examples of ecosystems that fail that definition, as much as several of the alternatives definitions for software ecosystems, because although they demonstrate actor interaction that results in software solutions or services, they are not structured on top of a common platform. Examples of such types of ecosystems emerged around OSGi, Open Design Alliance, or BitTorrent.

The lack of technological platform in ecosystems has been recognized as well by Jansen and Cusumano in their survey on software ecosystems [8] where they identify that a type of "underpinning technology" for software ecosystems can also be a standard apart from a (service) platform. Similarly, Manikas and Hansen [9], examine the Danish telemedicine ecosystem as a software ecosystem although the lack of a common technological platform identifying that the ecosystem under study demonstrates symbiotic relationships in actor and software level, motivated by a set of business models, and resulting in software products or services. Knodel et al. [10] report on an example of smart ecosystems in the agricultural domain based on a standard without a common platform. Thus, the concept of software ecosystems is evolving and we perceive the need to redefine the borders of software ecosystems. In this study we focus mainly on the common software (in particular the common technological platform) and reveal that there are different types of software ecosystems that do not necessarily include a common platform (at least in the traditional sense of a software platform).

[1] For instance, [2,3,4,5,6,7], please note that the list is not complete.

3 Ecosystem Building Blocks

We propose the meta-model of generic ecosystem building blocks depicted in Fig. 1 as the basis of our subsequent analysis of ecosystem types. The meta-model has been derived on the one hand from the analysis of existing literature and on the other hand from observations made in software ecosystems in practice. The building blocks are the following:

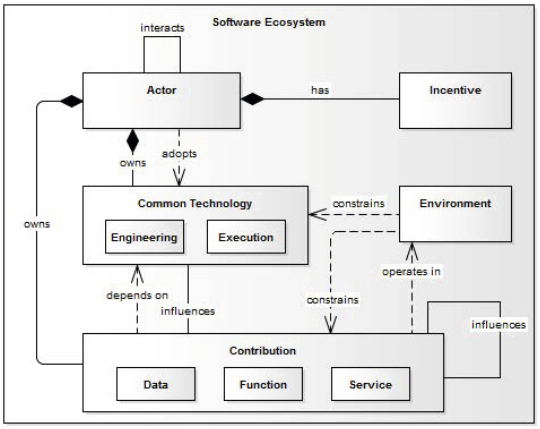

Fig. 1. Metamodel of ecosystem building blocks

- **Actor:** Ecosystems are driven by multiple actors interacting directly or indirectly with each other in collaborative or competitive nature. Actors provide a contribution to the ecosystem thus, the union of all contributions constitutes the moving target "ecosystem continuum". The number of actors is directly dependent on how open the ecosystem is to new actors, i.e. the entry barriers to the ecosystem. Typical instances of actors of an ecosystem may be individuals (developers, contributors, users, customer), commercial organizations, governmental entities, non-profit associations, and social communities.
- **Incentive:** Actors pursue some kind of incentive, which motivates their participation in the ecosystem. Typical instances comprise personal or business interest, fame, legal or standard regulations, legal or commercial forces, shared market needs or requirements.
- **Common Technology:** Ecosystems emerge around a shared technology. Instances of this technical linchpin can be twofold: (1) at engineering time (e.g., infrastructure, IDEs, SDKs, APIs, or standards) or (2) at execution time while the ecosystem is in operation (RTEs, platforms, frameworks, or protocols).
- **Contribution:** Actors provide contributions (with the linchpin being a special contribution as it is the key enabler of the ecosystem). Typical contribution may be software (functionality in form of apps, software service, or

stand-alone solutions; data in its raw form, aggregations, or context information) or services (management, integration, customization, etc.).
- **Environment:** The environment of the software ecosystem can be physical (interacting with the real world) or digital (IT only). It sets constraints for the software it is operating. Constraints may be imposed by special hardware, physical laws, social rules, or legal policies.

4 Analysis of Ecosystem Types

In this section we present different types of software ecosystems identified while discuss their characteristics according to the software ecosystem building blocks.

- **Cornerstone Ecosystems**
  Cornerstones are the more "traditional" types of software ecosystems: actors develop contributions on top of a common software platform typically extending the platform's functionality. Thus the existence of a technological platform is of central importance for the ecosystem of this type. The literature provides a number of examples of ecosystems of this kind like the iPhone AppStore, Android, or Eclipse. Cornerstone ecosystems and different perspectives of ecosystems of this type have been in focus of the research so far.
- **Standard-based Ecosystems**
  Compliance to standards is the key requirement for contribution in this kind of ecosystem. The standards replace a common technological platform and provide rather a specification of desired and required behavior of contributions, independent from their concrete realization as long as compliant. Standard-based ecosystem was initially proposed by Jansen and Cusumano [8]. Ecosystem standards usually are maintained and evolved organized by consortia with (paid) memberships. Standards often define rules to guarantee certain non-functional properties across individual contributions (e.g., safety in the ISOBUS standard in agricultural domain).
- **Protocol-based Ecosystems**
  Protocols are a less restrictive and more flexible technical linchpin of ecosystems. They provide a predefined specification of interaction of contributions with each other (e.g., exchange of data, call of software services).
- **Infrastructure-based Ecosystems**
  Infrastructure-based ecosystems share the same technical environment or tools at development time, but at the same time they provide independent contribution (e.g., Gnome, Github). The interactions between actors across individual contributions are often on a social level. Contributors themselves share their output and dedicate their efforts towards more than just one contributions (e.g., see [6] or [11]).

Table 1. Analysis of software ecosystem types

Category	**Cornerstone**	**Standard**	**Protocol**	**Infrastructure**
Emergence	successful product or actor(s)	specifications	need for use	successful product
Leadership	often run by single organization	often run by consortia	often run by community or organization	often run by (open source) community or company
Structure (Execution)	centralized, close collaboration, platform provides for governance, common technology part of the product	high level of actor & product independence, commitment to specific version	actor & product independence	different products, common technology not part of the product
Structure (Engineering)	cornerstone SDK shared across all actors	specification shared across all actors	API shared across all actors	common technology shared across all actors
Governance (common technology)	monarchic or aristocratic decisions about products (few decide, others have to follow)	federal decisions (no one can't do anything without shared agreement of all (key) parties)	democratic decisions (anyone can do anything, as long as the majority agrees	monarchic, democratic, or federal decisions about shared infrastructure
Governance (contribution)	obey the integrator (threat of being overruled)	stick to the rules	follow the guidelines	freedom of choice (anything possible)
Changeability (common technology)	orchestration dependent	slow, common agreement, backwards compatible	slow, common agreement, backwards compatible	orchestration dependent
Change Adoption (contribution)	orchestration dependent	painless (as long compliant)	painless (as long compliant)	independent of common technology

5 Discussion

The four types of software ecosystems are our starting point towards a typification of software ecosystems. In Table 1 we present the initial results on analyzing the major differences among the four types. In future work we aim at formalizing and extending the analysis and as well as adding a comparison to classical software product development outside an ecosystem.

We believe that the typification of software ecosystems must consider two distinct viewpoints: engineering and execution. Depending on its type, ecosystem expose different characteristics in their structure, governance, and the adoption of changes. Further, the leadership and emergence are key differentiators of ecosystem types. Based on these findings we argue that software ecosystem research has to adopt a broader view. In particular, the commonalities and specialties of each type should be analyzed to push software ecosystem research forward.

Goal of our future work is to come up with a well-defined taxonomy of software ecosystems and their characteristics. The taxonomy shall serve to guide researchers to focus on open challenges on the one hand and practitioners to learn from typical patterns and anti-patterns when participating in a software ecosystem on the other.

References

1. Messerschmitt, D., Szyperski, C.: Software ecosystem: understanding an indispensable technology and industry. MIT Press Books 1 (2003)
2. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: A research agenda for software ecosystems. In: 31st International Conference on Software Engineering - Companion, ICSE-Companion 2009, vol. 2009, pp. 187–190 (May 2009)
3. Bosch, J.: From software product lines to software ecosystems. In: Proceedings of the 13th International Software Product Line Conference, SPLC 2009, Pittsburgh, PA, USA, pp. 111–119. Carnegie Mellon University (2009)
4. Bosch, J., Bosch-Sijtsema, P.M.: Softwares product lines, global development and ecosystems: Collaboration in software engineering. In: Mistrik, I., van der Hoek, A., Grundy, J., Whitehead, J. (eds.) Collaborative Software Engineering, pp. 77–92. Springer, Heidelberg (2010), doi: 10.1007/978-3-642-10294-3_4
5. Bosch, J., Bosch-Sijtsema, P.: From integration to composition: On the impact of software product lines, global development and ecosystems. Journal of Systems and Software 83(1), 67–76 (2010)
6. Lungu, M., Lanza, M., Gîrba, T., Robbes, R.: The small project observatory: Visualizing software ecosystems. Science of Computer Programming 75(4), 264–275 (2010); Experimental Software and Toolkits (EST 3): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT 2008)
7. Manikas, K., Hansen, K.M.: Software ecosystems – A systematic literature review. Journal of Systems and Software 86(5), 1294–1306 (2013)
8. Jansen, S., Cusumano, M.A.: Software ecosystems – analyzing and managing business netowrks in the software industry. In: Jansen, S., Brinkkemper, S., Cusumano, M.A. (eds.) Software Ecosystems – Analyzing and Managing Business Netowrks in the Software Industry, pp. 13–28. Edward Elgar, Cheltenham (2013)
9. Manikas, K., Hansen, K.M.: Characterizing the danish telemedicine ecosystem: Making sense of actor relationships. In: Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems, MEDES 2013, pp. 211–218 (2013)
10. Knodel, J., Naab, M., Rost, D.: Supporting architects in mastering the complexity of open software ecosystems. In: Proceedings of the 2014 European Conference on Software Architecture Workshops, ECSAW 2014, pp. 1–13. ACM, New York (2014)
11. Mens, M.G.T., Goeminne, M.: Analysing ecosystems for open source software developer communities. Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry. Edward Elgar (2013)