Winter 12-2-2011

Making Sense Of Software Ecosystems: A Critical Review

Karthik Jayaraman

MAKING SENSE OF SOFTWARE ECOSYSTEMS: A CRITICAL REVIEW

Karthik Jayaraman, University of Oslo, karthikj@ifi.uio.no

ABSTRACT

Visualizing software as ecosystems has been an emergent phenomenon. The objective of this paper is to analyze the field of software ecosystems (SECO) and provide a critical review of the existing literature. This research identifies domains and peripheries of a SECO; highlights architectural challenges; examines design and control mechanisms and discusses some of the learning's from other popular paradigms that can be applied to address the key challenges in the SECO paradigm. This paper also aims to recommend future research directions for software ecosystems and its role in the broader context of information systems research.

Keywords: Software Ecosystems, Digital Ecosystems, Complexity, Information System Theory, Commons Based Peer Production, Open Source, Open Innovation

1. INTRODUCTION

Charles Darwin describes in his seminal work, Of the Origin of Species [1], that it is not the strongest of species that survive, nor the most intelligent, but the ones most adaptable to change. Similar to natural ecosystems that are complex and continuously evolving, the construction and management of software has become ever more complex and subject to continuous change. The survival of software systems is contingent to Lehman's software evolution law [2], which stipulates that a program must be continually adapted else it becomes progressively less satisfactory over time.

Complexity in the development and maintenance of software in terms of lines of code, functionality and interactions with other systems has continuously increased over the decades. As described by Bosch [3], the introduction of software product line approach and software engineering methodologies in the development of software has helped in dealing with this complexity and in streamlining the production of software. In today's rapidly evolving market place, companies have taken their product line architectures and components and made them available to parties external to the company. Bosch suggests that once a company decides to make its platform available to entities outside the organizational boundary, the company transitions to a software ecosystem. Companies in asymmetric competition in the market place use software ecosystems as a strategic tool to compete. An example of this fact is the asymmetric competition between Firefox and Internet explorer; although Mozilla foundation, which manages Firefox has lesser organizational resources compared to Microsoft, it has been able to successfully compete against a large well-established player [8].

True to Lehman's software evolution law, Firefox was able to adapt to the long tail needs of its customers by creating an open platform and providing developer friendly tools through which application developers created a plethora of Firefox apps and customizations.

This adaptation of Firefox to a changing industry environment enabled the survival and growth of both the ecosystem around Firefox and the browser. The fall of Nokia's Symbian and the recent success of the Apple iOS and Google Android have further fueled the need for the study of the ecosystem approach towards building successful software products [28].
Toffler describes consumers as a phenomenon of the industrial age and proclaims a shift to the Prosumer Age in which people produce many of their own goods and services [6]. Hippel conforms to Toffler's ideas of innovating in the Prosumer age and suggests that involving lead users of a product in the innovation process greatly enhances the attractiveness of the innovation [5][29].

Chesbrough describes that a firm can and should use external ideas as well as internal ideas, and internal and external paths to market, as they look to advance their technology [4]. One of the key roles of software ecosystems in the Prosumer age is to facilitate innovation and collaboration between producers and consumers of digital good and service [3].

Section 2 of this research identifies the domains and peripheries of a software ecosystem. Section 3 examines design and control challenges for companies and product developers in a SECO environment.

Section 4 compares SECO with other popular paradigms and looks at building upon existing theories to address challenges in a SECO. Section 5 summarizes the findings and provides future directions for the field of SECO. The contribution of this research is a critical review of the emerging literature around software ecosystems and suggestions of future research directions for this new paradigm.

2. Domains and peripheries

Researchers have taken different approaches towards describing the role of software ecosystems. The difference in perception of ecosystem peripheries and domains is primarily dictated by the author's object of study, which the author uses as a point of reference to view the ecosystem. From the existing literature, three main points of reference have been identified namely software ecosystems as organizational interactions, software ecosystems as a new layer of abstraction in technology platform construction and software ecosystems as strategic business and economic systems for growth in the market place.

2.1 Software ecosystems as organizational interactions

Bosch focuses on the organizational challenges that are addressed by software ecosystems and examines the inter-organizational interactions of a software ecosystem. Bosch suggests various benefit that this approach provides, which are increased attractiveness for new users, increased "stickiness" of the application platform, accelerated innovation through open innovation in the ecosystem and decreased TCO for creating new functionality by sharing the cost of maintenance with ecosystem partners [1]. Bosch further provides taxonomy to visualize the peripheries of ecosystems based on categories and platforms. Categories describe the methods of engaging with the ecosystem, which can be applications, operating systems or languages. The platform describes the interfaces, which can be based on web, desktop or mobile technologies.

Software ecosystems are modeled as multilevel interactions between organizations by Janesen et al [7]. The software ecosystem level encompasses the vendor level and the software supply network level. The software vendor level represents the organization that designs, builds, and releases software functionality within the SECO, the software supply network level represents buyers and suppliers who interact with the software vendor. The actors at various levels in the SECO function as a unit and interact with a shared market for software and services. Janesen further describes that the relationship between the actors at various levels in the ecosystem is based on a common technology platform or market space and the interactions are based on the exchange of information, resources and artifacts [9].

Messerschmitt et al identify software ecosystems as a group of businesses units working together and interacting with a shared market for software and services, these business units develop relationships among them. These relationships are formed due to the entities interest in the co-evolution of a common technological platform [27].

McGregor defines the ecosystem for a software product line as all the entities that interact with the product line organization. Information, artifacts, customers, money and products move among these entities as part of the planning, development, and deployment processes [11]. The product line ecosystem is a subset of the IT ecosystem, which is the large network of firms that drive the delivery of information technology products and services.

The view of software ecosystems as organizational interactions proposed by Bosch, Janesen, Messerschmitt and McGregor highlights the various tensions that arise from these interactions and describe the need for further study of these tensions.

2.2 Software ecosystems as a new layer of abstraction in the construction of technology platforms

Lungu et al describe software ecosystems as a collection of software projects; these projects are developed together and co-evolve in the same environment [12]. The environments in which software ecosystems evolve are classified into physical and virtual. Physical ecosystems represent the boundaries of the firm and the virtual ecosystems are the online communities of end users and developers.

Petra et al describe software ecosystems as a composition of a software platform, a set of internal and external developers, a community of domain experts and a community of users that compose relevant technical solution elements to satisfy their needs [13].

Akin to Petra et al, Goeminne et al describe the ecosystem as the source code combined with the user and developer communities that surround the software and highlight the need for the study of various technical components and interactions to understand how software evolves over time in the ecosystem [32].

Dhungana et al compare ecosystem processes in nature to the technical construction of software ecosystems. Learning's from evolution and bio diversity in nature demonstrate that the way to ensure development and sustenance of a software product ecosystem is to support a wide range of programming languages, platforms, hardware devices and a broad user community across domains and user groups [14].

Akin to Dhungana et al, Briscoe et al describe ways to learn from the self-organizing properties of nature for the construction of software ecosystems. The learning's from biological ecosystems found in nature can be translated to build robust, scalable digtal architectures, which can adapt to deal with complex, dynamic problems [33]. Briscoe et al further state that technologies such as Multi-Agent Systems (MASs), Service-Oriented Architectures (SOAs), and distributed evolutionary computing (DEC) can provide the foundation for assimilating the properties found in nature to construct complex digital ecosystems.

The technological view of software ecosystems highlighted by Lungu, Petra, Goeminne, Dhungana and Briscoe can assist stakeholders by improving re-usability, enhancing change management processes in software projects and in the evolution of better architecture for constructing complex ultra large-scale systems.

2.3 Software ecosystems as strategic business and economic systems for growth in the market place.

Levien describes an ecosystem in terms of value creation and identifies a business ecosystem as a subset of a software ecosystem [15]. Various roles in a business ecosystem exist namely keystones, dominators and niche players. A keystone creates and shares value with the rest of the ecosystem; a dominator seeks to extract as much value from the ecosystem, consequently destroying it. Niche players act to develop or enhance specialized capabilities that differentiate it from other firms in the network, leveraging resources from the network while occupying only a narrow part of the network itself.

Hence software ecosystems as a strategic tool should address the needs of various stakeholders in the ecosystem for its long-term sustenance. Iyer highlights the role of ecosystems in strategy formulation in organizations and describes two main roles of a keystone strategy; the first is to create value within the ecosystem and the second is to share the value with other participants in the ecosystem. Unless a keystone finds a way of doing this efficiently, it will fail to attract or retain members [16]. Hannesen identifies one of the strategic objectives for an organization developing an ecosystem as that of encouraging its customers to participate actively in the ecosystem [17]. The key driver for customers of a product line to participate in the evolution of a project is the ability to affect the development in ways that would benefit that customer. Hannesen further describes that this type of collaboration evolves into a self-regulating system where the product line developer (keystone) and the end customer who is actively engaged in the ecosystem mutually adapt to each other's strategic needs thereby leading to a win-win scenario.

Van den Berk et al describe the role of ecosystems in vision formulation and overall strategy of an organization through an ecosystem strategy assessment model [18]. Software ecosystems as a tool for vision and strategy building enables niche players to plan and orchestrate a future state of the software ecosystem. Van den Berk et al describe that a well-defined software ecosystem vision motivates partners to join the ecosystem and increases the opportunities for success for all the stakeholders involved in that ecosystem.

The business view of ecosystems as described by Levien, Iyer, Hannesen and Van den Berk demonstrate the strategic advantages of using ecosystems as a tool to compete in the market place and its role in facilitating development of new competencies through leveraging the resource of ecosystem partners.

3. Design and Control

Control in the development of software in a non-ecosystem context is left to the entities within the organization. In the software ecosystem context the product line transforms into a platform for developers external to the boundaries of the organization. This metamorphosis from product lines to ecosystems causes various challenges in the design and ways of controlling the evolution of the end product.

Bosch describes various architectural challenges in using a software ecosystem approach in the construction of products [19]. The keystone or the product controller should provide a stable interface between the platform and the external developers for collaboration and smoother integration of the product, security and control mechanisms should be used in the architecture to reduce the chances of a hostile entity introducing defective or malicious external code.

Hence the architecture needs to be designed for changes and future changes should be announced before the release for all actors in the ecosystem to co-ordinate.

One of the key problems when it comes to developing in an ecosystem is managing dependencies between various projects and developer groups. Dependencies can be managed by mapping the usage of various components and the impact of the changes to the components on the overall ecosystem. Hence understanding dependencies between the entities of a system is crucial in the design of scalable robust systems and Lungu et al describes that understanding these complex interdependencies between projects in an ecosystem is possible through documenting the overall ecosystem structure and more importantly making this knowledge available to the various developers of the ecosystem [20]. Monitoring the usage of various frameworks and the evolution of API's through which the components interact with the ecosystem reduces the chance of errors during integration. Charting frameworks that various projects use between each other enhances reuse of components and resources between projects. Thus improving the overall reliability and efficiency of the ecosystem.

The control of software ecosystems and their revenue models are determined through licensing of products and components that are part of the ecosystem. Software ecosystems are often made up of heterogeneously licensed products and components. These software licenses both facilitate and constrain the evolution of the ecosystem depending on their design. Hence the charting of licenses of various components is crucial in assisting the co evolution of various components and their interdependence [21].

Herold et al describe that due to complexity, life-cycle, and globalization issues that are predominant in ultra large scale projects such as software ecosystems, classical engineering approaches are no longer applicable. Software ecosystems cannot be planned, designed and implemented as a whole. To deal with these new challenges that are put forth by the emerging field of software ecosystems new software system engineering approaches are required [31].

The study of these new engineering approaches could form the future research agenda for the field of software ecosystems.

4. Building upon existing paradigms

Some of the key challenges in the emerging paradigm of software ecosystems are similar to the existing problems in the field of organizational design, architecture and the governance of digital commons, which have been addressed through existing fields of research such as complex adaptive systems, information infrastructures, commons based peer production and innovation studies.

The issues in the realm of Information Infrastructures & Complex Adaptive Systems revolve around the tensions between control and generativity in the design and architecture of systems. The drivers of generativity, which are change and control, are relevant to the emerging field of software ecosystems. Hanseth et al highlight the various challenges in designing and architecting digital infrastructures among which the bootstrap problem and the adaptability problem are highly relevant to software ecosystems.

The bootstrap problem deals with early users' needs in order to be initiated to participate in the system and influence its evolution; and the adaptability problem deals with the need for local designs to recognize information infrastructures unbounded scale and functional uncertainty [22], the challenges highlighted by Hanseth et al apply to software ecosystems and can be extended to solve similar challenges in the ecosystem context.

Tilson et al describe the paradoxical nature of digital infrastructures and debates the arbitrages between designing for stability versus flexibility and centralized versus decentralized control [23].

Tilson, Sorensen & Lyttnen further state the need to study the ways in which infrastructural change shapes IT governance, IS development, and promotes new effects across all levels of analysis, which are relevant to the study of emerging field of software ecosystems.

The recent discussions in the field of Commons Based Peer Production revolve around licensing, control points and the governance of communities. Hippel describes that the greater the involvement of the lead users of a product, greater is the resulting innovation [18]; hence the involvement of the end users and developer communities of a product holds the key to its innovativeness.

But as described by Chengalur-Smith et al, the attractiveness of an open source project is related to its governance structures. Licenses are a way to enforce the rights to use, copy and modify; the design of governance structures enables the right to gain visibility, to influence and to create derivatives of a project, whether in the form of spin-offs, applications or devices. More importantly, the governance model describes the control points used in governing platform boundaries and is a key determinant in the success or failure of an open platform [24].

The study of governance structures and incentives and their role in attracting platform developers, end users and application developers to various software ecosystems is a topic that applies to the emerging field of software ecosystems [30]. In the field of Innovation studies, the topics that can be applied to the study of software ecosystems are the subset of open innovation (Chesbrough 2003), user driven innovation (Hippel 2005) and digital innovation (Henfridsson 2009) [25]. One of the key debates within the field of innovation studies is the tension between exploration and exploitation (March 1991) and how companies manage this tension [26].

Firms can implement the process of exploration through open innovation, where firms establish ties with other organizations to share each other's knowledge and mutually benefit from the resulting innovation. But as companies venture out to partner with external vendors, end users and developer communities by opening up their platform through controlled modularity: various tensions are brought about in the management of the innovation network and in the sharing of the created value [30].

Existing research on how firms that provide product platform control them and how these controls evolve over time is limited, Henfridsson et al describe the need for a study of platforms that are based on the layered modular architecture. Software ecosystems are designed on a layered modular architecture and hence the issues mentioned by Henfridsson et al are relevant to the study of software ecosystems. Some of the existing research on organizational interactions and control points in open innovation networks can be applied to the field of software ecosystems to address some of the key challenges of organizational design and governance of stakeholder interactions in software ecosystems.

5. Conclusions & Future directions

This research has identified some of the emerging debates in the field of software ecosystems. Although different authors perceive software ecosystems as belonging to various domains and having different peripheries, a key pattern is observable across literature.

Most of the authors concur the role of a software ecosystem as an architectural tool to manage complexity or as a strategic tool to maximize value by leveraging resources external to the boundaries of the firm.

Software ecosystems provide a means to learn from processes in nature by assimilating them in the construction of software and in the design of organizations [14]. The key strength of software ecosystems, which is participation of external actors and open component interactions, is also its key challenge unless appropriate design and control mechanisms are formulated.

As ecosystems are comprised of various projects and complex loosely defined components that interact with each other; too much control in an ecosystem leads to users moving to other ecosystems and too little control would lead to bugs or other inefficiencies in the end product, hence an optimal control mechanism that promotes developers to partake in a SECO activity is crucial to its success.

The study of control mechanisms in complex socio-technical settings warrants detailed investigation and could play a key role in the future research agenda in the field of software ecosystems.

As described by researchers, the field of software ecosystems compliments the existing body of literature on the study of socio-technical interactions.

This new paradigm brings about some key challenges in the form of organizational, technical and social tensions.

Some of these tensions are addressed by existing information systems theories such as complex adaptive systems, digital innovation and commons based peer production, which can act as a theoretical lens for understanding these tensions.

But new theories need to be evolved to address some of the challenges that are unique to the paradigm of software ecosystems. The study of these tensions and formation of new theories to address emerging challenges in the ecosystem context could form the future research agenda of the field of software ecosystems.

REFERENCES

[1] Darwin, Ch.: "Of the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life". Paleontological Society 1854. The Origin of Species, Chapter 4, London 1872; Everyman's Library, London: Dentand Sons 1967

[2]Lehman, M., "Laws of Software Evolution Revisited," in European Workshop on Software Process Technology. Springer,1996, pp. 108–124.

[3]Bosch, J. From software product lines to software ecosystems. In Proceedings ofthe 13th International Conference on Software Product Lines (SPLC). Springer LNCS, 2009.

[4]Chesbrough, H. Open Innovation: The New Imperative For Creating and Profiting from Technology, Boston: Harvard Business School Press,2003.

[5] Hippel, E. Democratizing Innovation. MIT Press, Cambridge, MA, April 2005.

[6] Toffler, A., The Third Wave, William Morrow.

[7] Jansen, S.; Finkelstein, A.; Brinkkemper, S. A sense of community: A research agenda for software ecosystems, Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. pp.187-190, 16-24 May 2009

[8] Wang, T., Wu L., and Lin Z. 2005. The revival of Mozilla in the browser war against Internet Explorer. In Proceedings of the 7th international conference on Electronic commerce (ICEC '05). ACM, New York, NY, USA, 159-166.

[9] Campbell P.R.J. A three-dimensional view of software ecosystems. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10), Carlos E. Cuesta (Ed.). ACM, New York, NY, USA, 81-84.

[10] Jansen, S., Finkelstein, A., and Brinkkemper, S. Business network management as a survival strategy: A tale of two software ecosystems. In Proceedings of the First Workshop on Software Ecosystems. CEUR-WS, vol. 505, (2009)

[11] McGregor, J. A method for analyzing software product line ecosystems. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10), Carlos E. Cuesta (Ed.). ACM, New York, NY, USA, 73-80.

[12]Lungu,M., Lanza, M., Robbes, T. The Small Project Observatory: Visualizing software ecosystems, Science of Computer Programming, Experimental Software and Toolkits (EST 3): A special issue of the Workshop on Academic Software Development Tools and Techniques Volume 75, Issue 4, April 2010, Pages 264-275, ISSN 0167-6423.

[13] Bosch-Sijtsema, P., Bosch, J. (2010) "From Integration to Composition: On the Impact of Software Product Lines, Global Development and Ecosystems," Journal of Systems and Software, 83 (1): 67-76.

[14]Dhungana, D., Groher, I., Schludermann, E., Biffl, S. 2010. Software ecosystems vs. natural ecosystems: learning from the ingenious mind of nature. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10), Carlos E. Cuesta (Ed.). ACM, New York, NY, USA, 96-102.

[15] Iansiti, M., Levien, R. (2004). The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability. Harvard Business School Press, 225p.

[16] Iyer, B., C.- H. Lee, et al. (2006). "Managing in a 'Small World Ecosystem': Lessons from the Software Sector." California Management Review. 48(3).

[17] Hanssen, G. A longitudinal case study of an emerging software ecosystem: Implications for practice and theory, Journal of Systems and Software, In Press, ISSN 0164-1212.

[18] Berk, I., Jansen, S., Luinenburg, L. Software ecosystems: a software ecosystem strategy assessment model. In Proceedings of ECSA Companion Volume'2010. pp.127~134

[19] Jan Bosch. 2010. Architecture challenges for software ecosystems. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10), Carlos E. Cuesta (Ed.). ACM, NY, USA, 93-95

[20] Lungu, M., Robbes, R., Lanza, M. Recovering inter-project dependencies in software ecosystems. In Proceedings of ASE'2010. pp.309~312

[21] T. A. Alspaugh, H. U. Asuncion, and W. Scacchi. The role of software licenses in open architecture ecosystems. In First International Workshop on Software Ecosystems (IWSECO-2009), pages 4--18, Sept. 2009.
[22] Hanseth, O., & Lyytinen, K. Design theory for dynamic complexity in information infrastructures: the case of building internet. Journal of Information Technology, 2010, 25(1), 1-19.
[23] Tilson, D., Lyytinen, K., and Sorensen, C. Research Commentary---Digital Infrastructures: The Missing IS Research Agenda. Info. Sys. Research 21, 4 (December 2010), 748-759
[24] Open Governance Index: Measuring the true openness of open source projects from Android to webkit , July 2011. http://www.visionmobile.com/rsc/researchreports/Open%20Governance%20Index%20%28VisionMobile%29.pdf
[25] Henfridsson, O., Y. Yoo, et al. (2009). "Path Creation in Digital Innovation: A Multi-Layered Dialectics Perspective." Sprouts: Working Papers on Information Systems 9(20).
[26] March, J.G. Exploration and exploitation in organizational learning. Organization Science Vol 2, 1991
[27] Messerschmitt, DG., Szyperski, C. Software Ecosystem: Understanding an Indispensable Technology and Industry. Cambridge, MA, USA: MIT Press. ISBN 0262134322.
[28] West, J., Wood, D. "Open to Complementors: Tradeoffs of Ecosystem Management in the Symbian Mobile Phone Platform," 1st Tilburg Conference on Innovation, Tilburg, Netherlands, June 2010
[29] West, J., Bogers, M. "Contrasting Innovation Creation and Commercialization within Open, User and Cumulative Innovation," Academy of Management conference, Technology and Innovation Management division, Montréal, August 10, 2010.
[30] West, J., Wood, D., "Open to Complementors: Tradeoffs of Ecosystem Management in the Symbian Mobile Phone Platform," 1st Tilburg Conference on Innovation, Tilburg, Netherlands, June 2010.
[31] Herold, S., Klus, H., Niebuhr, D., Rausch, A. Engineering of IT ecosystems: design of ultra-large-scale software-intensive systems. (ULSSIS '08). ACM, New York, NY, USA.
[32] Goeminne, M., Mens, T., A framework for analysing and visualising open source software ecosystems. In Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE) (IWPSE-EVOL '10). ACM, New York, NY, USA, 42-47
[33] Briscoe, G., Wilde, DP. Computing of applied digital ecosystems. In Proceedings of the International Conference on Management of Emergent Digital EcoSystems (MEDES '09). ACM, New York, NY, USA, , Article 5 , 8 pages