Software Ecosystem: Features, Benefits and Challenges

J.V. Joshua, D.O. Alao, S.O. Okolie, O. Awodele
Department of Computer Science, School of Computing and Engineering Sciences, Babcock University, Ilishan-Remo, Ogun State, Nigeria.

Abstract—**Software Ecosystem (SECO) is a new and rapidly evolving phenomenon in the field of software engineering. It is an approach through which many variables can resolve complex relationships among companies in the software industry. SECOs are gaining importance with the advent of the Google Android, Apple iOS, Microsoft and Salesforce.com ecosystems. It is a co-innovation approach by developers, software organisations, and third parties that share common interest in the development of the software technology. There are limited researches that have been done on SECOs hence researchers and practitioners are still eager to elucidate this concept.**

A systematic study was undertaken to present a review of software ecosystems to address the features, benefits and challenges of SECOs.

This paper showed that open source development model and innovative process development were key features of SECOs and the main challenges of SECOs were security, evolution management and infrastructure tools for fostering interaction. Finally SECOs fostered co-innovation, increased attractiveness for new players and decreased costs

***Keywords**—Software ecosystem; Open source; closed system*

I. INTRODUCTION

The notion of ecosystems originates from ecology. One definition in Wikipedia defines an ecosystem as a natural unit consisting of all plants, animals and micro-organisms (biotic factors) in an area functioning together with all of the non-living physical (abiotic factors) of the environment.

Although the above is an excellent definition, it is less suitable here and therefore we start from the notion of human ecosystems. A human ecosystem consists of actors, the connections between the actors, the activities by these actors and the transactions along these connections concerning physical or non-physical factors.

Software ecosystems (SECO) refer to the set of businesses and their interrelationships in a common software product or service market [9]. A Software Ecosystem consists of the set of software solutions that enable, support and automate the activities and transactions by the actors in the associated social or business ecosystem and the organizations that provide these solutions [1].

This is an emergent field inspired in concepts from and business and biological ecosystems [14].

Well known examples of communities that may be seen as software ecosystems are Apples iPhone, Microsoft, Google Android, Symbian, Ruby and Eclipse.

Ecosystem concept may refer to a wide range of configurations. Yet, they all involve two fundamental concepts: a network of organisations or actors, and a common interest in the development and use of a central software technology.

The software industry is constantly evolving and is currently undergoing rapid changes. Not only are products and technologies evolving quickly, many innovative companies are experimenting with new business models, leading occasionally to fundamental shifts in entire industry structures and how firms and customers interrelate[17]. Recently, many companies have adopted the strategy of using a platform to attract a mass following of software developers as well as end-users, building entire “software ecosystems” (SECOs) around themselves, even as the business world and the research community are still attempting to get a better understanding of the phenomenon.

This paper explores the main terms under consideration which are the meaning of SECO, identify the main features of Software Ecosystems (SECOs) and finally establish the benefits and challenges of SECOs

II. WHAT IS THE PROBLEM

In the past few decades, we have witnessed different types of software development methodologies ranging from waterfall, spiral, component, chaos, rapid application development, rational unified process to agile models respectively. Almost all the models mentioned encourage development of software product entirely on the organisation concerned.

The emergent of Software Ecosystem (SECO) development paradigm has brought about co-innovation as a result of different players, however research communities and practitioners are still grasping to understand this concept. Hence this work is aim to expose what is known about software ecosystems (SECOs).

III. OBJECTIVES OF THE STUDY

The goal of the study is to carry out a systematic study of software ecosystems in order to present a wider view of what is currently known about software ecosystems

The specific objectives are to:

a) Identify the main features of Software Ecosystems (SECOs).

b) Establish the benefits and challenges of SECOs

IV. Scope Of The Study

It is not easy to study existing Software Ecosystems (SECOs) due to the fact that many SECOs are closed communities and it is hard to get access to information. Therefore, we adopted free open software ecosystems as our subject of studies.

V. Significant Of The Study

The significance of the study is to create awareness about the emergent fields of software ecosystems for research communities and practitioners and to establish research direction for software ecosystems.

VI. Review Of Related Research

Bosch [1] proposed a Software Ecosystem (SECO) taxonomy that identifies nine potential classes of the central software technology as shown in Table1 below, according to classification within two broad dimensions. The first one is the category dimension, which ranges from operating systems to applications, and to end-user programming. The second one is the platform dimension, ranging from desktop to web, and to mobile.

TABLE I. SOFTWARE ECOSYSTEM TAXONOMY

end-user programming	MS Excel, Mathematical,VHDL	Yahoo!Pipes, Microsoft PopFly, Google's mashup editor	none so far
Application	MS Office	SalesForce, eBay, Amazon, Ning	none so far
operating system	MS Windows, Linux, Apple OS X	Google AppEngine, Yahoo developer, Coghead, Bungee Labs	Nokia s60, Palm, Android, iPhone
category platform	Desktop	Web	Mobile

In Software Engineering (SE) community, studies of SECOs were motivated by the software product lines (SPLs) approach aiming at allowing external developers to contribute to hitherto closed platforms [1].

[4], opined that a potential benefit of being a member of a software ecosystem is the opportunity to exploit open innovation an approach derived from open source software (OSS) processes where actors openly collaborate to achieve local and global benefits. External actors and the effort they put into the ecosystem may result in innovations being beneficial not only to themselves (and their customers) but also to the keystone organisation, as this may be a very efficient way of extending and improving the central software technology as well as increasing the number of users.

According to [8] closer relationships between the organisations in an ecosystem may enable and improve active engagement of various stakeholders in the development of the central software technology.

When explaining the concept of software ecosystems it is also necessary to address how software ecosystems relate to the development of open source software [6]. There are clear similarities between these two concepts, but also several differences, which justify the definition of software ecosystems as a unique concept. The main difference between these two relates to the underlying business model. [3], explain the open-source business model as follows: *"The basic premise of an open-source approach is that by "giving away" part of the Company's intellectual property, you receive the benefits of access to a much larger Market. These users then become the source of additions and enhancements to the product to increase its value, and become the target for a range of revenue-generating products and services associated with the product."*

Whereas in a closed software ecosystem the intellectual property (the code) is *not* shared in any way.

However, different research directions indicated by literature and industrial cases re-enforce a lot of important perspectives to be explored, such as architecture, social networks, modelling, business, mobile platforms and organizational-based management [9]. Besides, SECOs involve a multidisciplinary perspective, including Sociology, Communication, Economy, Business and Law. These studies are also motivated by the software vendors' routine since they no longer function as independent units that can deliver separate products, but have become dependent on other software vendors for vital software components and infrastructures such as operating systems, libraries, component stores, and platforms [2].

VII. Architecture Of Major Software Ecosystems (Secos)

1) Symbian Software Ecosystem

In this ecosystem as shown in figure 1, the different categories of licenses and partner relationships included are as shown:

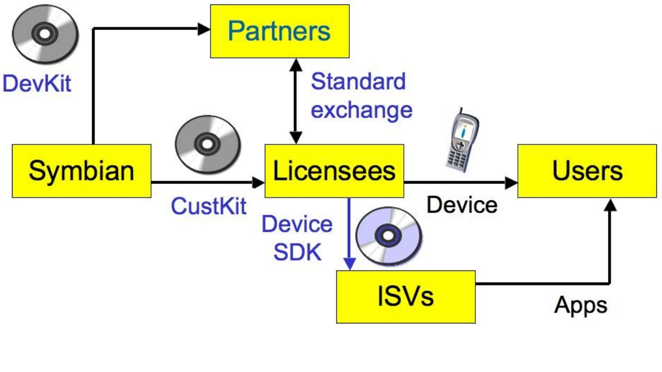

Fig. 1. Symbian Ecosystem [16]

Symbian described its network of customers and complementors as an “ecosystem”,

In the Symbian ecosystem, the different categories of licenses and partner relationships included are:

- System integrators or “licensees” (handset manufacturers) that integrated externally sourced software and internally developed hardware to create new devices (i.e. handsets) for sale to end users.
- CPU vendors worked to ensure Symbian OS compatibility with their latest processors.
- User Interface companies.
- Other software developers sometimes referred to as independent software vendors (ISVs) including developers of user applications and also middleware components such as databases.
- Network Operators, which in most countries were the dominant distribution channel for phones, and also decided what software components were preloaded on phones.
- Enterprise software developers, for cases where a company developed Symbian compatible software for its employees that use Symbian phones.

In many cases, members of Symbian’s ecosystem were also members of competing mobile phone ecosystems, such as those surrounding the Palm OS, Windows Mobile, and later Linux based platforms such as the LiMo Foundation and Google’s Open Handset Alliance (Android).

2) *Microsoft Software Ecosystem (SECO)*

Microsoft ecosystem consists of the following components: Device manufacturers, Independent Software Vendors (ISVs), Value Added Resellers (VARs), Office Equipment Dealers and Systems Integrators (SI) as shown in (Figure 2), and can all benefit from working together. But rarely do the ecosystem pieces remain static. New software applications are consistently being rolled out. And the VARs, dealers and SIs that sell and support these systems change with them.

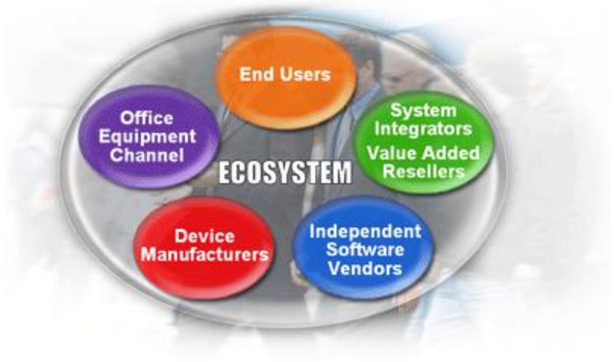

Fig. 2. Microsoft Software Ecosystem [7]

Microsoft sit at the centre of ecosystem. Ecosystems are an essential ingredient in delivering customer-focused solutions. And they help drive standards. And, they present revenue opportunities for all the partners involved. It’s no wonder that Microsoft spends so much money on building their ecosystem

The Microsoft ecosystem of applications, partners, and highly skilled IT resources provides customers with the best choice.

3) *iPhone Software Ecosystem*

The iPhone ecosystem which is one of the Apple’s three sub-ecosystems consists of the following components

- Developers and Designers
- Distribution
- Devices
- Users
- Internet
- Services and Advertisers

iPhone components are shown in figure3 below.

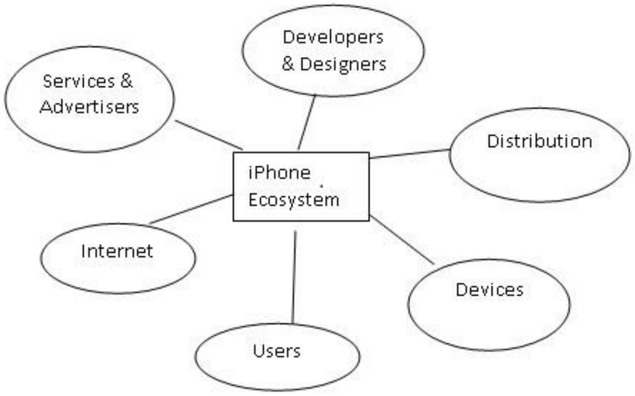

Fig. 3. iPhone components

Developers designs and implement complex interfaces smoothly and efficiently on limited hardware. C++ and

Objective-C are the primary languages used. Apple has historically put very little effort into supporting developers and designers, but has stepped up efforts for the iPhone platform. Designers are crucial to the success of iPhone applications. Developers simply utilise various technologies available to give designers what they want and need to build excellent interfaces.

4) *Ruby Software Ecosystem*

Ruby is a dynamic, open source programming language with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write. It was created by Yukihiru Matsumota in 1995 in Japan.

The Ruby Software Ecosystem consists mainly of two elements i.e. Gems and Developers with possible relationships among them. If a developer has a relationship with a gem, he is a developer of that specific gem.

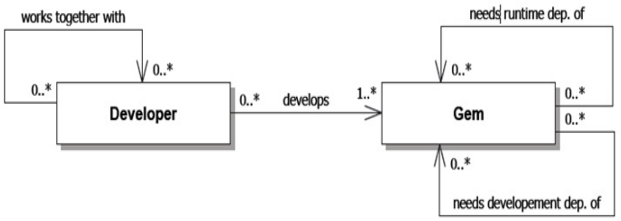

Fig. 4. Ruby Software Ecosystem [11]

The entire Ruby ecosystem consists of all developers, gems and their relationships as shown in figure 4. Some corporate high technology initiatives with Ruby are: **Sun Microsystems**, **Microsoft**, **Apple, IBM and SAP**.

5) Google Android Ecosystem

Android is a comprehensive open source platform designed for mobile devices. It is championed by Google and owned by Open Handset Alliance. The open Handset Alliance prominent members include: T-Mobile, Motorola, Samsung, Sonny Ericsson, Toshiba, Vodafone, Google, Intel, and Texas instrument. This list has grown multi fold with over 80 in number [5].

Android is revolutionizing the mobile space. It is a truly open platform that separates the hardware from the software that runs on it. This allows for a much larger number of devices to run the same applications and creates a much richer ecosystem for developers and consumers.

One way in which Android is quite different from other platforms is the distribution of its applications. On most other platforms, such as iPhone, a single vendor holds a monopoly over the distribution of applications. On Android, there are many different stores, or markets. Each market has its own set of policies with respect to what is allowed, how the revenue is split, and so on. As such, Android is much more of a free market space in which vendors compete for business. The figure 5 below summarised android software stack.

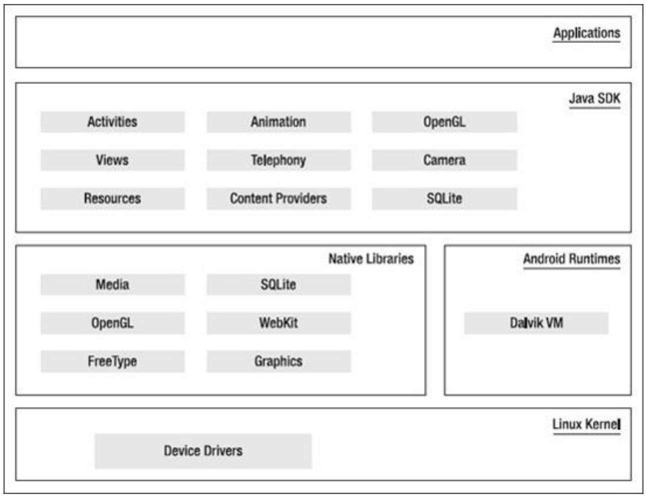

Fig. 5. Android Software Stack [13]

6) Eclipse Ecosystem

Eclipse is an open source integrated development environment (IDE) for Java. It was originally aimed to provide a united platform for different IDE products from IBM.

The Eclipse project, which began at the end of 1998, has an ambition to "eclipse" the leader of the IDE market. Within few years, Eclipse has evolved from Java IDE (version 1.0) to a universal tooling platform (version 2.0), and finally evolves to an application framework for building rich client application (version 3.0). Commercial software development tools such as IBM Rational tool, web sphere studio, and Borland JBuilder have been developed based on Eclipse.

Eclipse is currently managed by the Eclipse foundation with over 100 members including HP, IBM, Nokia, INTEL and Borland. The biggest challenge for the foundation is to cope with its rapid growth from its community.

Eclipse ecosystem Architecture

The functional building blocks of the Eclipse IDE are illustrated in Figure 6 below. The entire platform is open source and royalty-free for other open source or commercial products that add new building blocks.

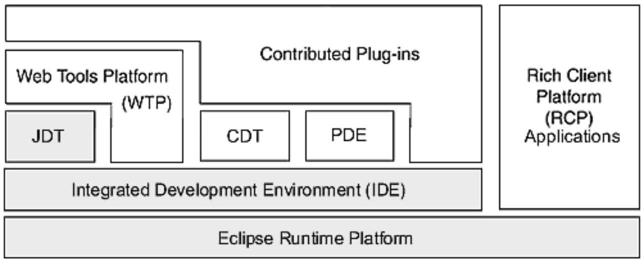
Contributed Plug-ins
Web Tools Platform (WTP)
JDT
CDT
PDE
Rich Client Platform (RCP) Applications
Integrated Development Environment (IDE)
Eclipse Runtime Platform

Fig. 6. Eclipse ecosystem Architecture [12]

A. Components of the Eclipse ecosystem Architecture

1. C/C++ Development Tools (CDT)

The C/C++ Development Tools (CDT) project is creating a fully functional C and C++ IDE for the Eclipse platform.

2. Plug-in Development Environment

The Plug-in Development Environment (PDE) supplies tools that automate the creation, manipulation, debugging, and deploying of plug-ins.

3. Java Development Tools

Java Development Tools (JDT) are the only programming language plug-ins included with the Eclipse SDK. However, other language tools are available or under development by Eclipse subprojects and plug-in contributors

4. Eclipse Runtime Platform

The core runtime platform provides the most basic level of services such as Loading plug-ins and managing a registry of available plug-ins, managing resources, update and help facility.

5. Integrated Development Environment

The Eclipse IDE provides a common user experience across multi-language and multi-role development activities.

6. Web Tools Platform

The mission of the Web Tools Platform (WTP) project is to provide a generic, extensible, and standards-based tool platform that builds on the Eclipse platform and other core Eclipse technologies.

7. Rich Client Platform

The Eclipse Rich Client Platform (RCP) is a set of plug-ins needed to build a rich client application.

The eclipse consortium is currently hosting eight top level projects and over thirty sub-level open source projects. There are also countless number of commercial and open source Eclipse related products, plug-ins, and distributions available from the internet. This virtual ecosystem takes care of software development, application life cycle, data management, and business operations

VIII. Open Source Software (Oss) And Closed Ecosystems - Similarities And Differences

TABLE II. The Similarities And Differences Between Open Source Software And Closed Systems

Similarities

A shared interest in the development, evolution, and use of a software product
Independent actors collaborate and contribute to development
Open innovation
New business models as compared to traditional licensed software

Differences

OSS	**Closed ecosystems**
Open source code.	Closed source code.
Ownership is shared.	Ownership and control lies with the keystone organisation.
Free use (with options for paying for specializations and related services)	Pay for use.
Extensibility through open source code.	Extensibility through controlled interfaces

IX. Features Of Software Ecosystems

The main features of SECOs are as follows.

1) They Inherits characteristics of natural ecosystems like mutualism, commensalism, symbiosis and so on

2) SECOs have architectural concepts like interface stability, evolution management, security and reliability

3) It is an to open source development model

4) They can be used to negotiate requirements for aligning needs with solutions, components, and portfolios

5) SECOs have capability for process innovation.

X. Benefits Of Software Ecosystems

1) Fosters the success of software co-evolution and innovation inside the organization involved and increases attractiveness for new players

2) Decreases costs involved in software development and distribution

3) Help analyse and understand software architecture

4) Supports cooperation and knowledge sharing among multiple and independent software vendors

5) Enables better analysis of requirements and communication among stakeholders

6) Help to overcome the challenges during design and maintenance of distributed applications

7) Provides help to the tasks of business identification, product architecture design and risk identification

8) Provides information for the product line manager regarding software dependencies

XI. Challenges Of Software Ecosystems

1) Establishing relationships between ecosystem actors and proposing an adequate representation of people and their knowledge in the ecosystem modelling.

2) Several key architectural challenges such as: platform interface stability, evolution, management, security, reliability.

3) Heterogeneity of software licenses and systems evolution in an ecosystem and how organizations must manage these issues in order to decrease risks of dependence.

4) Companies have difficulty at establishing a set of resources in order to differentiate from competitors.

5) Technical and socio-organizational barriers for coordination and communication of requirements in geographically distributed projects.

6) Insufficient infrastructures and tools for fostering social interaction, decision-making and development across organizations involved in both open source and proprietary ecosystems.

XII. CONTRIBUTIONS

This paper contributes to the field of software ecosystems by providing

1) A necessary foundation for understanding how Software Ecosystems are composed and further aids understanding of this new and expanding area of software development.

2) A number of open research questions and challenges which should enable scholars interested in SECOs to swiftly gain an overview of this research area

XIII. FUTURE DIRECTIONS FOR SOFTWARE ECOSYSTEMS

As with most novel approaches, this paper on SECO has opened up possibilities for new and exciting future directions. This following area should be investigated as future research directions/challenges for SECOs.

1) In Open source ecosystems.

a) How can quality be measured per developer?

b) How can relationships be formed between developers?

c) How can conflicts be resolved in open source ecosystems?

d) How can application program interfaces (APIs) to third-party components be used.

2) Governance.

a) What are the best strategies for survival in an ecosystem?

b) How can organisations involved achieve and maintain a healthy position in a SECO?

3) Analysis

a) How can an ecosystem be analysed.

b) Is it possible to create models, visualizations, and large data sets for analysis?

4) Openness

Every software platform at the centre of an ecosystem has to have some degree of openness. The main research question here is

How can openness in software affects and influences the success of a business, where there appears to be a real trade-off between the height of entry barriers and number of third parties willing to participate in the ecosystem.

5) Quality

a) How can ecosystems deliver the highest quality experience to customers in the ecosystem?

b) What are measures that participants can take to increase quality?

XIV. CONCLUSION

This paper provides a review of SECOs and confirmed that it is an emergent field that has been mainly inspired by studies from business and natural ecosystems. We highlighted that SECOs field needs more industrial studies to increase its body of evidence. Also, given the current state of research and practice in SECOs, we envisaged the need to conduct integrative studies among research communities and industry.

Finally the paper proposes a number of open research questions and challenges to enable scholars interested in SECOs to swiftly gain an overview of the research area and to help them in their own research endeavours.

REFERENCES

[1] Bosch, J. (2009). From Software Product Lines to Software Ecosystems. In proceedings of 13th International Software Product Line Conference (SPLC'09), San Francisco, USA, 24-28 August.111-119.

[2] Boucharas, V., Jansen, S., and Brinkkemper, S., (2009), 'Formalizing Software Ecosystem Modeling'. In: Proceedings of the 1st International Workshop on Software Ecosystems, 11th International Conference on Software Reuse, Falls Church, USA, 34-48, September.

[3] Brown, A. W. and Booch, G. (2002). *Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors. In proceedings of 7th International Conference on Software Reuse:* Methods, Techniques, and Tools, Austin, USA, April 15-19. 123-136.

[4] Chesbrough, H. (2006). Open Innovation: A New Paradigm for Understanding Industrial Innovation. In Open Innovation: Researching a New Paradigm. Chesbrough, H., Vanhaverbeke, W. and West, J. (eds.). Oxford: Oxford University Press: 1-12.

[5] Fabio Cevasco (2011) Ruby Compendium: An essential Guide to the Ruby Ecosystem.

[6] Fitzgerald, B. (2006). The Transformation of Open Source Software. MIS Quarterly **30**(3): 587-598.

[7] Gantz J.F, Bibby D. (2011) White paper on Partner Opportunity in the Microsoft Ecosystem.

[8] Hanssen, G.K. and T.E. Fægri,(2008) Process Fusion -- Agile Product Line Engineering: an Industrial Case Study. Journal of Systems and Software **81**: p. 843---854

[9] Jansen, S., Brinkkemper S., Finkelstein A. Bosch J.(2009), Introduction to the Proceedings of the First Workshop on Software Ecosystems, in First International Workshop on Software Ecosystems. CEUR--WS.

[10] Jansen S., Brinkkemper S., Finkelstein, A.(2009) A Sense of community: A research agenda for software ecosystems. In: Proceedings of the 31st International Conference on Software Engineering.

[11] Kabbedijk, J., and Jansen, S., (2011), 'Steering Insight: An exploration of the Ruby Software Ecosystem'. In: Proceedings of the 2nd International Conference on Software Business, Brussels, Belgium, 44-55, June.

[12] Lam T., Gotz A. (2005)' Leveraging The Eclipse Ecosystem for Scientific Community'10th ICALEPCS Int. Conf. on Accelerator & Large Expt. Physics Control Systems. Geneva, 10 - 14 Oct 2005, TH3A.3-5O (2005)

[13] Mark Gargenta (2011) Learning Android: O'Reilly media Inc.

[14] Moore, J. F. (1993). Predators and prey: A new ecology of competition. Harvard Business Review 71(3): 75-86.

[15] Wirehead Labs, Inc. (2012). The iPhone Ecosystem

[16] Wood, David (2002). "Symbian Developer Expo 2002 - in context" internal presentation,Symbian Ltd., London.

[17] Xu, L., Brinkkemper, S. (2007): Concepts of product software. European Journal of Information systems 531-541