Defining Software Ecosystems: A Survey of Software Platforms and Business Network Governance

Slinger Jansen[1,2] and Michael Cusumano[2]

[1] Utrecht University, Utrecht, the Netherlands,
Slinger.jansen@uu.nl,
[2] Massachusetts Institute of Technology, Cambridge, Massachusetts,
{Slinger,Cusumano}@mit.edu

Abstract. Currently, there is little understanding about how different types of software ecosystems must be governed for the preservation and improvement of ecosystem health. This paper explores the definition of software ecosystems and provides a classification model for software ecosystems. The classification model is applied to 19 cases previously explored in software ecosystem literature, and governance tools are observed for the different types of ecosystems. The governance tools are summarized in a governance model that, when used correctly, serves ecosystem coordinators in determining strategies to maintain and ultimately improve software ecosystem health.

1 Introduction

While in the early days of software engineering a software product was the result of effort of an *independent* software vendor to create a monolithic product, modern software strongly relies on components and infrastructure from third party vendors or open source suppliers [11, 36, 10]. The relationships between software development firms and service companies shaped the product software landscape into software ecosystems, where suppliers and buyers of software products, components and technologies collaboratively create competitive value. One could state that the success of a product software company therefore no longer depends only on its own development quality but also on the way it manages its relationships [16, 22, 15]. Software differs from physical goods in several ways. Software has no physical limitation, therefore the main limitations are conceptual, social and economical [5, 31]. No other business has a gross profit margin of 99 percent on sold products [11]. In other words, reproduction costs for software products are next to zero.

Just as participants in a value chain of physical products, partners in a software value chain maintain ongoing business relationships. Up to the late 80s, vertically integrated companies delivered complete system stacks [11]. These stacks contained everything needed to serve a customer; hardware and software; operating system and applications. In the late 80s and beginning of the

90s the horizontal layer structure of solution stacks changed into more modular clusters [23]. The 'software stack' is now split up in activity layers that are complimentary to each other [16, 17] through interfaces and middleware. Because of this market structure, it is not uncommon that two software producing organizations may collaborate on one activity level and be in competition on another.

At the highest level of abstraction, software ecosystems are collections of organizations that are related through software or a software related concept. Please also note that software ecosystems are subsets of business ecosystems. If we were to take all organizations in the world and define queries on it to create specific collections of organizations, we could for instance use the query "Toyota". This would provide us with all the suppliers, partners, customers, and resellers that do business with Toyata and their relationships. The collection of organizations and relations gives us access to the business ecosystem of this particular company, i.e., the Toyota ecosystem. The same can be done for software concepts, to get the software scope instead of the generic business scope. A non-exhaustive list of software concepts is provided with examples:

- **Standards -** XML, BPM, OSGi, J2EE, Corba, SEPA, etc.
- **Products -** OpenOffice, Microsoft Word, SAP BusinessOne, Grand Theft Auto, etc.
- **Hardware -** Playstation 3, HTC Diamond, PDAs, BMW 5 series, etc.
- **Platforms -** .Net, Facebook, Android, OS X, etc.

The list can be used to identify types of ecosystems (again, non-exhaustive) and to name specific instances, such as the "OS X Ecosystem". Several notes must be made with this list. The first issue concerns inclusion and exclusion criteria or, put differently: at what point does a collection of organizations and relations become a business ecosystem instead of a software ecosystem. The answer to this question can be made complex, but it can be simplified by determining the aim of the research or "query" that is done within a specific business ecosystem. If the query is not specifically software related ('What companies use SEPA for all their business transactions and how are these companies related?') the resulting set is a business ecosystem. If the query is software related ('What software providers have already adopted SEPA as the main language for their APIs and how do these organizations relate to each other?') the resulting set can be used to compose a software ecosystem.

A second issue with software ecosystems is the type of participant. In a commercial ecosystem participants are easy to determine: they are typically organizations that aim to survive and thrive, whether it is a one-man company or a large ecosystem orchestrator, such as SAP, Microsoft, or Facebook. In the scope of open source ecosystems participants can range from foundations (e.g., the Eclipse Foundation), to commercial organizations (e.g., RedHat), to independent developers (e.g., David Heinemeier Hanson for the Rails ecosystem). The question on what constitutes a participant in a software ecosystem depends on the following criteria: is it relevant for the research question, is it contributing

to the ecosystem in a meaningful (but not necessarily significant) way, and is the contribution software related?

One question that is often asked is whether customers are part of the software ecosystem. If we draw the analogy to biological ecosystems, customers are the 'plankton' that keep the ecosystem alive and well, which by definition includes them into the ecosystem. A smart query can be created, however, that excludes customers altogether, such 'what organizations build Apps for Android and how are they related to each other?' The extended answer to the question if customers are part of the ecosystem thus is: it depends on the query. The term 'plankton' in this context is highly appropriate: without a (potential) market of sufficient size it is risky for third-parties to join or co-create an ecosystem.

The second challenge that is handled in this work is the definition of a series of governance tools for platform leaders, i.e., parties that can determine the future of the ecosystem. Software ecosystem governance is defined as procedures and processes by which a company controls, changes or maintains its current and future position in a software ecosystem [1]. With this governance model platform leaders can determine whether they are using the right methods for their ecosystem governance to reach their strategic goals.

This paper continues with a description of the survey research in Section 2. In Section 3 several terms related to software ecosystems, such as biological, digital, and business ecosystems are discussed and contrasted against software ecosystems to provide perspectives on the definition of software ecosystems. Section 4 provides the table with the survey results of this work and a classification model is extracted to classify and categorize software ecosystems by their defining characteristics. Four types of ecosystems are further studied in Section 5, where different governance tools are identified through the survey of ecosystems and summarized in the software ecosystem governance model for health preservation and improvement, after we finalize with a short discussion and conclusion in Section 6.

2 Research Approach

The main research question of this research is whether there exist common types of software ecosystems and whether these ecosystems have similar factors in the domain of governance, such as life threats, growth challenges, who is in control, etc. To identify and select software ecosystems, two criteria were used: first the definition of software ecosystems was followed closely, and if an ecosystem did not adhere to this definition, it was not selected. Secondly, the ecosystems were selected from papers as described in common software ecosystem literature [4], to make sure that these ecosystems have also been identified by others as being relevant software ecosystems.

The survey was stopped at the current number, because of time constraints and for the sake of brevity. We therefore do not claim completeness: the contribution of this work lies in the first classification of software ecosystems and in the identification of common concerns for the governance of these ecosystems.

For each ecosystem a list of 16 questions was answered by performing document and web site study. Each of the ecosystems was checked by at least one other researcher for correctness. These sixteen questions were extracted from the work on open software enterprises [27] and the work on software ecosystem governance [1]. Based on the answers found for each of the surveyed software ecosystems, statements were identified that enable someone to swiftly classify a software ecosystem.

3 Background

In this section the comparison with other kinds of ecosystems is made. We start with the comparison with biological ecosystems and then move to the supercategories of ecosystems that can contain software ecosystems: business ecosystems and digital ecosystems. We finalize with a definition of software ecosystems.

3.1 Biological Ecosystems

Considering the software ecosystems domain "borrows" part of its name from the domain of (biological) ecosystems, there are some relationships between the two, as others have established [14, 21]. Iansiti and Levien highlight the example of the jaguar, which is known to eat up to 85 species, thereby controlling large parts of the ecosystem, even though the elegant and nimble jaguar only takes up a minute part of the ecosystem in total body mass. This could be compared to a platform leader like Microsoft, which, although seemingly large with its 100,000 employees, only is a minute part (less than 1%) of the ecosystem in terms of developers and revenues.

Dhungana et al. [14] go quite far in their comparison. In short, they equate software and natural ecosystems on the following issues. First, both types of ecosystems have a finite reservoir of resources. Secondly, participants in both ecosystems will be included or forced out by changes in the dynamics in the ecosystem. Thirdly, collaboration and competition occur in both types of ecosystems. Finally, there are life cycles in both ecosystems (product versus organism).

The comparison to biological and natural ecosystem is easily made, but analogies only stretch so far. The main difference between software and natural ecosystems is that biological ecosystems are mainly studied to observe influences from external factors, whereas software ecosystem dynamics are analysed mainly with the aim of growth and success. Software ecosystems are also made up of participants harboring intentionality, whereas the beings in a biological ecosystem have no means to consciously be part of the ecosystem. The largest difference between participants in software ecosystems and those in natural ecosystems, however, is that in software ecosystems participants can consciously decide to exit the ecosystem or even destroy it.

3.2 Business Ecosystems

Software ecosystems are perhaps a new concept, but business ecosystems have been around much longer. Since the 90s, James F. Moore has used the concept in multiple outlets, but most famously in his book entitled "The Death of Competition" [33]. Moore defines business ecosystems as an economic community supported by a foundation of interacting organizations and individuals: the organisms of the business world. The economic community produces goods and services of value to customers, who are themselves members of the ecosystem. The member organisms also include suppliers, lead producers, competitors, and other stakeholders. Over time, they coevolve their capabilities and roles in supply chains, and tend to align themselves with the directions set by one or more central companies. Those companies holding leadership roles may change over time, but the function of ecosystem leader is valued by the community because it enables members to move toward shared visions to align their investments, and to find mutually supportive roles [32].

It must be noted that the concept has primarily found ground in the technical community. Furthermore, Moore is already referring here to "central companies", a concept that is later reformulated to "platform leaders" by Cumano and Gawer [19]. It is commonly found that using business ecosystems to quickly develop, prototype, and release new products has a positive influence on innovation, since it is much quicker than more traditional "in-house only" product development processes [12].

The notion of platform leaders quickly leads to the typical three roles that are mentioned in contexts of business ecosystems: platform leaders, niche players, and bridge players. platform leaders are typically orchestrators that largely determine the growth of an ecosystem. A specific type of platform leader is the dominator, a platform leader that quickly swallows up large parts of the ecosystem, such as IBM did in the 80s, and more recently Twitter, that, by doing a number of strategic acquisitions, made new entrants into the Twitter ecosystem reluctant to join. The niche players are those parties that use technology from the platform leaders to approach certain niche markets and are typically smaller, but still highly successful. An example of a niche player in a software ecosystem is for instance a provider of technical drawing software for heating in buildings, built on top of AutoCAD. The adagium 'get big, get niche, or get out' is a popular rewording of the platform leader-niche player phenomenon. Finally, the bridge player is a player that bridges certain ecosystems. An example of such a player in a number of software ecosystems is PhoneGap, a platform that enables the release of software applications in different mobile ecosystems with one version of the software code.

Software ecosystems have been the subject of much debate before the concept was defined formally. They have always been seen as yet-another-instance of business ecosystems and have been studied in that way with much merit [20, 34, 30]. In the previous section it has been explained that software, however, is different and therefore deserves its own subcategory under business ecosystems.

3.3 Digital Ecosystems

A digital ecosystem is defined as a distributed adaptive open socio-technical system with properties of self-organisation, scalability and sustainability [9]. The field of digital ecosystems is rapidly evolving. Previously, digital ecosystems were described to be a form of evolution from traditional monolithic and service-oriented architectures towards collaborative architectures in which autonomous agents, orchestration, and service choreography are the main topics addressed. Increasingly, however, the term digital ecosystem is being used in policy and research to describe digital *business* ecosystems. For sake of simplicity, we consider software ecosystems subsets of digital ecosystems.

3.4 Software Ecosystems

Several scholars conducted in-depth studies to explain ties between companies in software ecosystems. These studies range from strategic level [17, 15, 8, 22] through operational level [28] down to the technical level [26, 25]. Scope changes of the interpretations of software ecosystems result in differing definitions as well. At present several different definitions exist of the term software ecosystems.

- Kittlaus and Clough [29] define a "software ecosystem as an informal network of (legally independent) units that have a positive influence on the economic success of a software product and benefit from it".
- Bosch [7] defines a software ecosystem as "consisting of the set of software solutions that enable, support, and automate the activities and transactions by the actors in the associated social or business ecosystems and the organizations that provide these solutions".

Three shared concepts stand out in these definitions: (1) actors, organizations and businesses, (2) networks and social or business ecosystems, and (3) software. Based on these shared concepts Jansen, Finkelstein, and Brinkkemper [24] constructed the definition that is used throughout this article:

> *A **software ecosystem** is a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts.*

Software ecosystems are usually governed and steered by one or more coordinating parties who profit when the ecosystem thrives. Typically, these coordinators also control the 'underpinning technology' on which the ecosystem is based, such as a commercial company that builds a software platform. There are also less traditional ecosystem coordinators, however, such as consortia behind open source platforms or even single developers who may have influence on the ecosystem. *Software ecosystem coordinators* are defined as beneficiaries of

software ecosystem growth who have instruments available to influence the development of the platform or the surrounding ecosystem. Typically they are also (partly) responsible for the further development of the underpinning technology.

The role of **software platforms** in software ecosystems is undeniable. If an ecosystem is to be formed around anything, the controlling entity needs to allow a degree of freedom for value creation of which the controlling entity receives only a small part [18, 19]. As Iansiti and Levine [21] articulate: "Outside complementors will be attracted to the platform if there is option value in the complements, provided the platform owner does not expropriate all the value they create." In this paper we use the platform definition of Gawer and Cusumano [19]: "A foundation technology or set of components used beyond a single firm and that brings multiple parties together for a common purpose or to solve a common problem". Furthermore, Gawer and Cusumano state that the value of the platform increases exponentially with (a) more complementary products and services, and (b) more users.

4 Survey and Classification Model

In table 4 the software ecosystems that were studied for this work are listed, along with the four classification factors. The classification factors were extracted from sixteen properties that were extracted from the work by Baars and Jansen [1] and the work on the Open Software Enterprise [27]. The sixteen properties were left out for reasons of brevity, however, factors were included such as a detailed description, the underlying base technology, the population, a typical example within the ecosystem, and the estimated size of the ecosystem in terms of number of platform extenders[1].

Typology and classification models have been created for business ecosystems. The model of Boons and Baas [6], for instance, identifies product life cycle, material life cycle, geographical area, sectoral, and miscellaneous ecosystems. Our classification, however, specifically looks at four factors, being:

Base Technology - The definition of a software ecosystem suggests that in general there will be some technology underpinning the ecosystem. The survey in this paper suggests that *all* software ecosystems are underpinned by a technology. The types of technology found here are a software platform, a software service platform, and a software standard. Most software ecosystems in the survey are underpinned by one software platform, such as AutoCAD, Ubuntu, or Wordpress. Some software ecosystems are even underpinned by multiple software platforms, such as the Microsoft ISV ecosystem, which is underpinned by Microsoft CRM, Sharepoint, Exchange, and many other underlying platforms. The third type of base technology for software ecosystems is a software service platform, where an online service is provided around which other ecosystem participants can gather, such as the Force.com platform and the HubSpot platform,

[1] The full survey can be found here:
http://www.softwareecosystems.org/empirical/governancesurvey/

Table 1. Ecosystems Under Study and their Classification according to the Ecosystem Classification Model

Name	Underpinning technology	Coordinators	Extension market	Accessibility
AutoCAD plug-ins	platform	privately owned	a list of extensions	paid
Ubuntu	platform	consortium	multiple extension markets	for free
Android	platform	privately owned	a commercial extension market	screened
iOS	platform	privately owned	a commercial extension market	paid
Eclipse	platform	consortium	an extension market	screened
XBMC	platform	consortium	multiple extension markets	for free
Joomla	platform	consortium	a list of extensions	screened
GX	platform	privately owned	an extension market	screened
Ruby	platform	consortium	multiple extension markets	for free
Ogre3D	platform	consortium	a list of extensions	screened
MS ISV Partners	platforms	privately owned	a commercial extension market	paid
Wordpress	service platform	consortium	an extension market	screened
HubSpot	service platform	privately owned	a commercial extension market	screened
SalesForce	service platform	privately owned	a commercial extension market	screened
Spotify	service platform	privately owned	an extension market	screened
World of Warcraft	service platform	privately owned	multiple extension markets	for free
XBRL	standard	consortium	a list of extensions	paid
Open Design All.	standard	consortium	a list of extensions	paid
OSGi	standard	consortium	a list of extensions	paid

which are only available online and cannot be installed individually on a customer's server. Finally, an ecosystem can be underpinned by a software standard, such as the XBRL ecosystem or the Open Design Alliance, which bases most of its technology around AutoDesk's DWG standards. Please note that some technologies are based on other technologies and therefore may have an overlap with those ecosystems. The case of Joomla, for instance, relies on a database, a web server, and an operating system and its participants are all also participant (sometimes unknowingly) in these other ecosystems. Furthermore, please also note that we abstract from the technology used for platform extension (i.e., API, service call, plug-ins, apps). For an extensive discussion on extension patterns the work of Jansen et al. [26] can be used as a starting point.

Coordinators - The coordinators of an ecosystem are perhaps the largest influence on governance: the owners have full control over the tools and methods that are used to increase the success of an ecosystem. A software ecosystem is either owned by a community or owned by a private party. An ecosystem that is controlled by a community is the Eclipse ecosystem. The Eclipse consortium controls the ecosystem and represents the wishes and commands of the consortium members, ideally. An ecosystem in which the technology is privately owned by a commercial party with typical commercial interests, usually exerts more control over the ecosystem. An example of a commercially controlled ecosystem is the iOS ecosystem, where Apple (the private owner of the underpinning technology) for instance determines what types of applications are accepted to the application store.

Extension Markets - Many of the software ecosystems are centralized around a market of extensions, also known as app stores or app markets. A software ecosystem can have no extension market, a simple list of extensions, an actual extension market, a comercial extension market, and multiple extension

markets. When a software ecosystem has no explicit extension market, components may be available through known third-parties. An example of this is the AutoCAD ecosystem, where there is a short list of components on the Autodesk community site, but there exist many more extensions available only through third-parties that do their own marketing. A more mature way of handling extensions is a list of extensions. The methods of getting on the list are determined by the *accessibility* of the extension market. An example of the extension list is provided on the site of the three dimensional graphics platform Ogre3d, where developer contributions to the ecosystem are listed on a simple web page. The next level of extension market actually enables parties to distribute and even sell their extensions on the market. An example of this is the Firefox Extensions market, where third parties can offer their extensions, without having to pay the Firefox consortium. A commercial extension market is used by the owner to make money, with the obvious examples being the Android App Market and the iOS App Stores. Finally, some ecosystems, such as the XMBC and World of Warcraft ecosystems, have multiple extension markets. In the case of the World of Warcraft software ecosystem, the maker of the game did not want to create their own add-on store and left it to the market to create solutions for that. Several development hubs and add-on markets have been created, such as "curse.com" and "wowace.com".

Accessibility - Accessibility to an ecosystem is one of the defining factors for an ecosystem: the ability to join an ecosystem and its barriers to entry determine what types of participants will play a part in it. For software ecosystems three accessibility possibilities have been identified: open source, screened but free, and paid. An open source ecosystem is one where it is possible to add contributions to a project, create and publish components in the extension market, etc., without any barriers. An example of such an open source ecosystem is the Rails ecosystem, where developers can simply add components (gems) to the community without anyone performing another check. Typically, however, a committee performs some quality control to make sure that a developer's contribution is of the right quality, such as with Eclipse Plug-ins and plug-ins for the Chrome browser. The most restricted is an ecosystem for which must be paid. An example of such an ecosystem is the iOS ecosystem, where a developer must pay a small amount before being able to publish applications in the extension market of Apple. When combining these four qualifiers the following defining sentence results:

> The [NAME] software ecosystem is based on a {*software platform, software service platform, software standard*} and is coordinated by a {*privately owned entity, community* } with {*no extension market, a list of extensions, an extension market, a commercial extension market, multiple extension markets*} to which participants can submit extensions {*for free, after a screening, after making a payment*}.

For example, the Apple iOS software ecosystem is described as follows:

> The Apple iOS software ecosystem is based on a *software platform* and coordinated by a *privately owned entity* with *a commercial extension market* to which participants can submit extensions *after making a payment*.

It must be noted that several qualifiers were excluded, but may play a defining role in describing a software ecosystem. One of those candidates was stickiness, based on the characteristics "network effects", "switching costs", and "multi-homing". The main problem with stickiness, i.e., how hard it is for participants in an ecosystem to switch to another, is that it can only be defined in continuous scales instead of discrete terms as the qualifiers do now. A second qualifier that was excluded was entry barriers, which are highly dynamic and can change frequently. Another factor that defines an ecosystem that was left out of the defining statement is the number of markets that the ecosystem brings together. Most software ecosystems concern only two markets, such as AutoCAD users and AutoCAD plug-in builders. For other ecosystems, however, this is much more complex, such as for a mobile ecosystem in which extension builders, phone users, hardware providers, phone service companies and a central platform coordinator are brought together in one ecosystem. Finally, the market size has been excluded because a considerable (potential) market is considered a prerequisite for any software ecosystem.

Observation 1: Standards are a special kind of ecosystem. When studying the data, some patterns can be discovered. To begin with, standards are an interesting foundation for an ecosystem: multi-homing (i.e., the use of multiple standards) is always possible, they are generally managed by consortia with paid memberships, and standards are usually promoted using showcases as opposed to an extension market for software platforms.

Observation 2: Market ownership tells the story. Typically, in case a private entity manages the platform the ownership of the market lies in the hands of that entity. As a consequence, when there are multiple extension markets the platform is typically managed by a community (with the exception perhaps, of Android, which has multiple extension markets and a commercial one). If the extension market is a "commercial" market, i.e., money can be made by selling an extension in the market, it is always owned by a private party. When an ecosystem has multiple extension markets, they are typically openly accessible and do not require a fee up front. When extenders need to pay for an extension market to get their extensions in there, switching costs are typically high.

Observation 3: Freely accessible ecosystems also allow multi-homing. All ecosystems encountered that do not have a paid membership or entry barrier, as a logical consequence typically also allow multi-homing.

Observation 4: Extension lists have relatively high entry barriers. When an ecosystem coordinator manages a list of extensions instead of an extension market, it is typically hard to get onto the list. For most community-managed ecosystems getting onto the list means the extension needs to be approved by the core team of developers. For privately managed ecosystems, it requires approval of the platform leader and frequently involves joining the partner program.

Observation 5: Service platforms are always in the hands of privately owned parties. Due to the running costs of an online service, online service platforms are always managed by commercial organizations. Even Wordpress.com, which is based on the open source content management platform Wordpress, is managed by Automattic, a commercial entity that contributes to the Wordpress community actively.

5 Governance Tools

Coordinators of software ecosystems have little insight into the governance tools that are available to them. In this section a governance model for ecosystem health preservation and improvement is presented, that provides ecosystem coordinators with a set of tools for maintaining and improving the health of the ecosystem, by stimulating growth, economic activity, and robustness of the ecosystem. For each of the ecosystem health aspects, being robustness, niche creation, and productivity, governance tools are provided. These governance tools are provided for four different kinds of ecosystem coordinators. Software ecosystem governance is defined as procedures and processes by which a company controls, changes or maintains its current and future position in a software ecosystem [1]. Although the definition applies to any company in the software ecosystem, we focus on the efforts that the "platform leaders" are directing at improving their position in the software ecosystem. Furthermore, the health measures of Iansiti and Levien [21] and the operationalization of Den Hartigh [13] are taken as the starting point for the governance methods, in the sense that we focus on robustness, niche creation, and productivity as the core of ecosystem health. Furthermore, the typology from the previous section is taken into account, to illustrate the difference between a community run software platform, a commercially run software platform, a community run standard, and a commercially run standard. The governance model for ecosystem health preservation is presented in figure 1 and is based on the results from the case studies, the Open Software Enterprise model [27], the governance model of Baars et al. [1], and the work of Den Hartigh et al. [13]. Please note that this work provides a top-down view for the governance of an ecosystem. For a more practical bottom-up approach one might consult the work "The Art of Community" [2], which provides concrete tools for creating, maintaining, and growing an open source community.

5.1 Software (Service) Platform Governance

Community-driven software platform - Community driven software platforms are typically large open source products, but sometimes, such as in the case of IntelliCAD, closed source products that are managed by a consortium. Coordinators of such communities have to deal with independent individuals and organizations that use the platform for their own means, and these coordinators have to make sure the platform and its development process remain

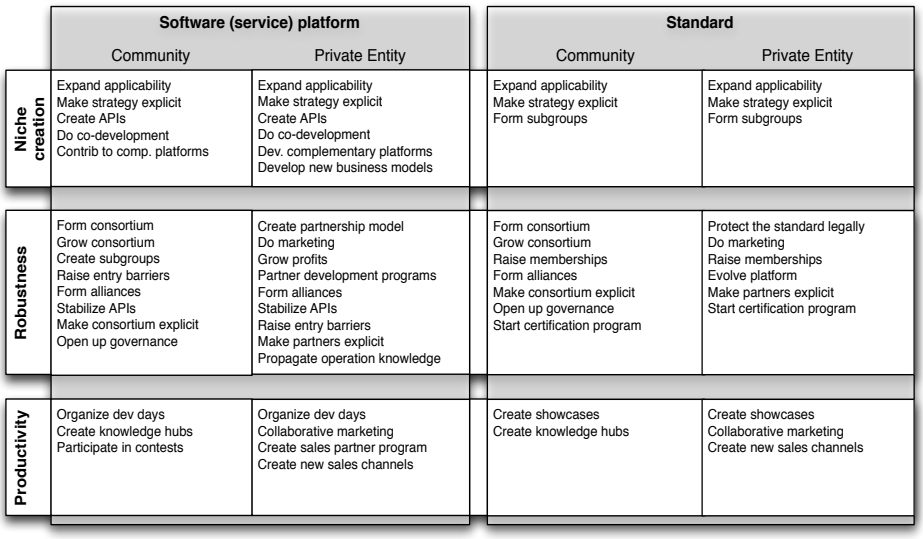

Fig. 1. Governance Model for Ecosystem Health Preservation and Improvement

active and healthy [35]. Coordinators have to make sure that sufficient *niche creation* is happening for other parties to join in. Steps that can be taken are the creation of APIs, the extension of the applicability of the platform (for instance by venturing into new domains), and the strategy of the platform must be made explicit. By making the strategy explicit, i.e., product lifecycle, acquisition strategy, platform strategy, and ecosystem strategy, niche players can rest assured that their position in the ecosystem will remain safe. Furthermore, community coordinators can do co-development and co-funding requests with third parties to attract them to the ecosystem. Finally, for niche creation, community coordinators can contribute to complimentary platforms, because as those complimentary platforms grow, so does the platform of the coordinators.

In regards to *robustness*, community coordinators must make sure the community remains as lively and stable as possible, to provide a strong core to which third parties can commit safely. In the beginning, community coordinators can form a community to formalize the way in which contributors and third-parties can assist in keeping the ecosystem healthy. Once the consortium is explicit, it can be grown by attracting new members. Furthermore, subgroups can be created to bring together domain specialists in an ecosystem, thereby further mobilizing the community itself. To make the ecosystem more stable, a community can further raise the barriers of entry, to create a critical mass of dedicated core members who are fully committed to the platform: in the case of the Eclipse platform several companies are providing millions of dollars in development for the platform, simply to be a strong steering member in the Eclipse ecosystem.

Next to that, ecosystem coordinators can form alliances with other ecosystems to create further complimentaries and subgroups within the ecosystem. Furthermore, by stabilizing the API, the platform can become a slowly evolving core of functionality for the community. Finally, the consortium must be made explicit to show the world who are key members of the organization. As a final step, the consortium can open up governance, to have the consortium govern itself through the years.

With respect to *productivity*, community managers have several tools available to make the ecosystem create more value. A typical tool to use is to raise awareness and activity surrounding the platform by organizing development days and by organizing and participating in contests that require the use of the platform. Finally, the creation of knowledge hubs enables participants in the community to share and find knowledge, thereby making them more productive and effective.

Privately owned software platform - Private owners of software platforms, such as Microsoft, Apple, and Autodesk, want to maximize their value by penetrating the market as deeply as possible. This penetration is reached mostly by associating with domain specific parties that leverage the platform within the ecosystem to create value for customers that would have never been reached without these domain experts (i.e., niche players). Coordinators of these platform-based ecosystems are similar to communities in that they too want to continuously increase the use of the platform, however, the main difference is that these coordinators want to maximize their profit as well. When looking at *niche creation*, all the same tools can be used as community coordinators do: expand the applicability of the platform, make the strategy explicit, create APIs, do co-development with partners, and develop or contribute to complimentary platforms. Coordinators of commercial platform ecosystems, however, can also create niches and business opportunities by introducing new business models for third parties. A relevant example is that of the commercial extension markets that are on the increase, which enable third parties to make profits from customers they previously could not reach.

To increase *robustness*, the private company can do several things to create stability and stimulate activity in the ecosystem. Typically, these start with the development of a partnership model that enables third parties to participate and create value in the ecosystem according to set roles and positions in the ecosystem. Furthermore, the general strengthening of the hub and ecosystem by doing marketing, raising prices, and growing profit stabilizes the ecosystem significantly (within limits). Partners with potential or weak partners can play a strong role in (de)stabilizing the ecosystem, so the introduction of a partner development program can strengthen weak participants and bring high potentials closer to the ecosystem. An example of this is when Microsoft pays for the development of new extensions in their Windows Mobile Application Market. Also, the forming of alliances with other hubs in ecosystems, the raising of entry barriers, and the propagation of software operation knowledge (i.e., how the software and its extensions run at customers) throughout the ecosystem can

raise quality and stability [37]. Raising entry barriers can increase the robustness of the ecosystem. By increasing membership fees, raising quality levels, starting certification programs, and assigning different levels within the partnership programs, the ecosystem will grow a stable core of committed members. Finally, the stabilizing of APIs creates consistency within the ecosystem and enables partners to create trustworthy and stable extensions to the platform.

With respect to *productivity*, coordinators of a privately owned platform can also organize development days to increase activity. Furthermore, collaborative marketing and sales can be done, to emphasize that smaller extending third parties have a respectable relationship with the platform leader. Finally, the creation of new sales channels to enable more revenue for the ecosystem participants can play a big part in productivity of the ecosystem.

5.2 Software Standard Governance

Software standards are different from software platforms in that they represent not a platform or a software artifact, but a set of standardized interfaces to enable communication and exchange of information across different organizations. These standards organizations guard the standard and grow the ecosystem around it. Governance of these ecosystems is different in several aspects: the core of the ecosystem is knowledge, not a software platform. It is highly uncommon for a privately owned organization to set a standard, however examples are the DWG format of Autodesk, previously Adobe's PDF format, and Microsoft's Office file formats. In the data set we did not find many commercial organizations managing a standard, so the fourth column in figure 1 is mostly based on literature.

Community Driven Standard - Most standards are driven by a thriving community of participating organizations, all with different aims with the standard but with a common goal. Examples of such organizations are W3C, ISO, and many others. To *create niches* for a standard, its applicability can be extended to include new domains. Furthermore, when a standards organization makes its strategy explicit to the community, participants become aware of why and how the standard will benefit them and whether they are in direct conflict with the community or not. Finally, the establishment of subgroups is beneficial for the creation of new niches in which the standard can be applied.

In regards to *robustness*, different tools can be applied to increase it, such as the formation of a formalized consortium, the growing of the community, or the raising of memberships to strengthen the community organization and ensure its continuation in the future. The formation of alliances with other (complimentary) standards or platforms may also be beneficial, as it encourages further adoption of the standard. Also, the community can open up the governance of the consortium, for instance by establishing how decisions within the consortium must be made. Finally, a certification program enables an ecosystem coordinator to raise quality standards and establish certain partners as being highly valuable for the ecosystem.

In regards to *productivity*, it is hard for the standards organization to determine tools for increasing adoption and use of the standard. However, two tools that are available and widely used are the use of showcases of how the standard has benefited the organization that uses it, and also the creation of knowledge hubs to inform participants from different domains about the use of the standard.

Privately Owned Software Standard - There are some occurrences of privately owned software standards, such as the Microsoft Office File Format and the DWG File Format of Autodesk, used for the exchange of the 3D drawings. The way in which these ecosystems are formed and stimulated are similar to those owned by a community or consortium, with the only exception that the standard is typically partly closed and protected by intellectual property laws, to leverage the advantages of owning a widely used standard, such as the sale of APIs and libraries for the reading of such libraries. In regards to *niche creation*, private coordinators can apply the same tools as communities: expand applicability, make the strategy of the standard explicit, and form subgroups.

To increate *robustness*, the standard can be legally protected by using user licenses and intellectual property law. Furthermore, robustness is determined by the strength of the standard, so raising usage fees and doing abundant marketing for it will increase market penetration and stability. Also, the evolution of the standard keeps the standard relevant and outmanoeuvres competition. Finally, the usage of a partner program and possibly partner certification, further stabilizes the ecosystem.

In regards to *productivity*, knowledge hubs must be created, showcases can be used to show the effect of using the platform, and collaborative marketing enables standards users to leverage their partnership with the standard ecosystem coordinator. Also, the ecosystem coordinator can assist by creating new sales channels, for instance by connecting niche players with potential customers in specific domains.

The governance model as presented leads from the results from the survey, and to a small extend from standards literature. The model does not, however, claim to be complete. To achieve completeness, a structured literature survey needs to be conducted, which is considered future work. The model can then to be extended with the coring and tipping methods of Cusumano and Gawer [18], Cusumano's "levers" [19], and standards governance methods [3, 35].

6 Discussion and Conclusion

This paper presents a model for classifying software ecosystems and hopes to present a de facto standard for presenting cases and survey results on software ecosystems. The model has been kept deliberately simple, but aims to illustrate the defining characteristics of software ecosystems. With this model software ecosystem researchers can quickly gain insight into the characteristics that define any particular ecosystem. The classification model has been used to identify four different classes of software ecosystems. For each of these classes governance tools

have been identified and presented in the software ecosystem governance model for health preservation and improvement.

At the current stage it is hard to claim completeness of both the classification model and the governance model for ecosystem health preservation and improvement. Part of the future work is to further evaluate the model with ecosystem coordinators and to perform a more extensive survey of software ecosystems for improved validation of the models. Now that a list of governance tools has been created, the applicability of the tools and the dependent situational factors must be determined. In the future we hope to create a software ecosystem governance maturity model that provides guidelines for improving software ecosystem governance, depending on the maturity of a software ecosystem and other situational factors.

References

[1] A. Baars and S. Jansen. A framework for software ecosystem governance. In *Proceedings of the Third International Conference on Sw Business 2012, Boston, MA, USA*, 2012.

[2] Jono Bacon. *The Art of Community - Building the New Age of Participation*. O'Reilly, 2009.

[3] Paul Bannerman and Liming Zhu. Standardization as a business ecosystem enabler. In *Proceedings of the International Workshop on Enabling Service Business Ecosystems (ESBE08)*, 2008.

[4] O. Barbosa and C. Alves. A systematic mapping study on software ecosystems. In *Proceedings of the 3rd Workshop on Sw Ecosystems*. http://ceur-ws.org/Vol-746/, 2011.

[5] B. Beizer. Software is different. *Annals of Software Engineering*, 10:293–310, 2000.

[6] FAA Boons and L.W. Baas. Types of industrial ecology: the problem of coordination. *Journal of Cleaner Production*, 5(1):79–86, 1997.

[7] J. Bosch. From software product lines to software ecosystems. In *Proc. of the 13th Int'l Sw Product Line Conf.*, pages 111–119. Carnegie Mellon University, 2009.

[8] Sjaak Brinkkemper, Ivo van Soest, and Slinger Jansen. *Information Systems Development*, chapter Modeling of Product Software Businesses: Investigation into Industry Product and Channel Typologies, pages 1–19. Springer US, 2009.

[9] G. Briscoe and P. De Wilde. Digital ecosystems: evolving service-orientated architectures. In *Proceedings of the 1st international conference on Bio inspired models of network, information and computing systems*, page 17. ACM, 2006.

[10] E. Carmel. Cycle time in packaged software firms. *Journal of Product Innovation Management*, 12(2):110–123, 1995.

[11] M.A. Cusumano. *The Business of Sofware: What Every Manager, Programmer and Entrepreneur Must Know to Succeed in Good Times and Bad*. Free Press, New York, NY, 2004.

[12] JB DeLong. Why the valley way is here to stay. *Fortune*, 141(11):36–37, 2000.

[13] Erik den Hartigh, Michiel Tol, and Wouter Visscher. The health measurement of a business ecosystem. In *Proceedings of the European Network on Chaos and Complexity Research and Management Practice Meeting*, 2006.

[14] D. Dhungana, I. Groher, E. Schludermann, and S. Biffl. Software ecosystems vs. natural ecosystems: learning from the ingenious mind of nature. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, pages 96–102. ACM, 2010.

[15] Barbara Farbey and Anothny Finkelstein. Software acquisition: a business strategy analysis. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, pages 67–83. IEEE Computer Society, August 2001.

[16] Lucia Gao and Bala Iyer. Analyzing complementarities using software stacks for software industry acquisitions. *J. Manage. Inf. Syst.*, 23(2):119–147, 2006.

[17] Lucia S. Gao and Bala Iyer. Partnerships between software firms: Is there value form complementarities? In *Proceedings of the 41st Hawaii International Conference on System Sciences*, 2008.

[18] A. Gawer. *Platforms, Markets and Innovation*. Edward Elgar, 2009.

[19] Annabelle Gawer Gawer and Michael A. Cusumano. *Platform Leadership*. Harvard Business School Press, Boston, MA, USA, 2002.

[20] B.A. Huberman. *The laws of the Web: Patterns in the ecology of information*. The MIT Press, 2003.

[21] Marco Iansiti and Roy Levien. Strategy as ecology. *Harvard Business Review*, 82(3):68–78, March 2004.

[22] B. Iyer, C.H. Lee, and N. Venkatraman. Managing in a 'small world ecosystem': Lessons from the software sector. *California Mgmt. Review*, 48(3):28–47, 2006.

[23] Bala Iyer, Chi-Hyon Lee, and David Dreyfus. Competing in the era of emergent architecture: The case of packaged software industry. *Hawaii International Conference on System Sciences*, 0:209b, 2007.

[24] S. Jansen, A. Finkelstein, and S. Brinkkemper. A sense of community: A research agenda for software ecosystems. 2009. In 31st International Conference on Software Engineering, New and Emerging Research Track.

[25] Slinger Jansen, Sjaak Brinkkemper, and Anthony Finkelstein. Component assembly mechanisms and relationship intimacy in a software supply network. 15th International Annual EurOMA Conference Special Interest Session on Software Supply Chains, 2008.

[26] Slinger Jansen, Sjaak Brinkkemper, Ivo Hunnik, and Cetin Demir. Pragmatic and opportunistic reuse in innovative start-up companies, 2008.

[27] Slinger Jansen, Sjaak Brinkkemper, and Lutzen Luinenburg. Shades of grey: Opening up a software producing organization with the open software enterprise model. *Journal of Systems and Software*, 2012.

[28] Slinger Jansen, Anthony Finkelstein, and Sjaak Brinkkemper. Providing transparency in the business of software: A modelling technique for software supply networks. In *Proceedings of the 8th IFIP Working Conference on Virtual Enterprises*, pages 677–686, 2007.

[29] Hans-Bernd Kittlaus and Peter N. Clough. *Software Product Management and Pricing: Key Success Factors for Software Organizations*. Springer-Verlag Berlin, Heidelberg, 2009.

[30] Y.R. Li. The technological roadmap of cisco's business ecosystem. *Technovation*, 29(5):379–386, 2009.

[31] David G. Messersmschitt and Clemens Szyperski. *Software Ecosystem: understanding an indispensable technology and industry*. The MIT Press, Cambridge, Massachusetts, London, England, 2003.

[32] James F. Moore. Predators and prey: A new ecology of competition. *Harvard Business Review*, 71(3):75–86, May 1993.

[33] James F. Moore. *The death of competition: Leadership and strategy in the age of business ecosystems*. HarperBusiness, New York, 1996.

[34] B.F. Nattrass and M. Altomare. *The natural step for business: Wealth, ecology and the evolutionary corporation*. New Society Pub, 1999.

[35] Siobhn OMahony and Fabrizio Ferraro. The emergence of governance in an open source community. *Academy of Management Journal*, 50(5):1079–1106, 2007.

[36] S. Sawyer. Packaged software: implications of the differences from custom approaches to software development. *European Journal of Information Systems*, 9(1):47–58, 2000.

[37] H. van der Schuur, S. Jansen, and S. Brinkkemper. The power of propagation: on the role of software operation knowledge within software ecosystems. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, pages 76–84. ACM, 2011.