The Architecture of Generativity in a Digital Ecosystem: A Network Biology Perspective[1]

Completed Research Paper

SungYong Um
Fox School of Business, Temple University, 1810 North 13th Street, Philadelphia, Pennsylvania 19122, USA
sungyong.um@temple.edu

Youngjin Yoo
Fox School of Business, Temple University, 1810 North 13th Street, Philadelphia, Pennsylvania 19122, USA
youngjin.yoo@temple.edu

Sunil Wattal
Fox School of Business, Temple University, 1810 North 13th Street, Philadelphia, Pennsylvania 19122, USA
swattal@temple.edu

Rob J. Kulathinal
Department of Biology, Temple University, 1990 North 12th Street, Philadelphia, Pennsylvania 19122, USA
robkulathinal@temple.edu

Bin Zhang
Fox School of Business, Temple University, 1810 North 13th Street, Philadelphia, Pennsylvania 19122, USA
bzhang@temple.edu

Abstract

Firms are increasingly relying on third-party developers to innovate by building open digital ecosystems. We draw on a network biology approach to explore the structural pattern of how individual modules in a digital ecosystem interact with one another to produce an ever-evolving ecosystem landscape. We perform an empirical study of WordPress.org, the world's largest blog service platform. Using text mining and network analyses, we extract API (Application Programming Interfaces) used in plug-ins developed by third party developers, characterizing each plug-in as a combination of APIs. We further construct a topological overlap plot of the WordPress ecosystem comprising of APIs that are used with each plug-in allowing us to examine the underlying structure of generativity in the WordPress ecosystem through a co-expression network of APIs. Our results show that APIs form a complex and dynamic ecosystem that follow a discernible pattern distinctive from known distributions, even though there is no central designer who coordinates the design of all plug-ins and overall growth of the entire

[1] Funding for this research provided in part by National Science Foundation (SES – 1261977 and VOSS-1120966) and CIFREG. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funders.

WordPress ecosystem. Our analysis shows that while APIs created by the focal firm play a central role in shaping the architecture of generativity of its ecosystem, some of the external APIs developed by other firms also play an important role in shaping the ecosystem, showing the distributed nature of the architecture of the generativity.

Keywords: open digital ecosystem, generativity. co-expression network, topologic overlap, landscape

Introduction

As products and services are becoming increasingly digitalized, firms can no longer simply focus on enhancing features and the quality of their products through their own innovation efforts (Yoo et al. 2010). Instead, many firms try to leverage the creativity of others by creating an open digital ecosystem (Tiwana et al. 2010). An open digital ecosystem consists of: (a) a platform which is designed and maintained by a focal firm and (b) a collection of third-party developed complementary products (also known as plug-ins). A platform includes a set of boundary resources such as Application Programming Interfaces (APIs) and Software Development Kit (SDK) that focal firms provides for third-party developers to facilitate their development efforts, together with a set of rules and norms that govern the behaviors of the participating firms (Ghazawneh et al. 2012a; Tiwana et al. 2010).

Past research on modularity has shown that firms use modules to enhance flexibility of their products through a "mix-and-match" strategy without redesigning the entire system (Garud et al. 1993; Sanchez et al. 1996). However, emerging open digital ecosystems with third-party developers challenge this traditional concept of modularity based on a fixed system boundary and fixed set of relationships among modules (Yoo et al. 2010). New digital ecosystems have much more malleable boundaries as they harness the distributed creativity among heterogeneous actors who bring diverse knowledge and technological resources in creating new plug-in (Yoo et al. 2012). In such open digital ecosystems, novel and unanticipated recombination of existing software code modules such as APIs often serve as the source of generative and unbounded innovations (Arthur 2009). Therefore, the generativity of an open digital ecosystem is fundamentally different from the flexibility of a closed modular system. While the flexibility of a closed modular system comes from *the variety of modules of the same kind (Baldwin et al. 2000; Sanchez et al. 1996)*, the generativity of an open ecosystem comes from *the variety of plug-ins of different kind (Yoo et al. 2012)*.

In this study, we seek to understand the underlying architecture of generativity that we observe in such open digital ecosystems. Generativity refers to "the capacity of technology to produce unprompted changes driven by large, varied, and uncoordinated participants" (Zittrain 2006). The logic of generativity offers, therefore, an insight on how organizations such as Apple and Google offer a large array of innovations through their platforms. However, little attention has been paid to explore the underlying structure of generativity and how innovations by some actors can instigate "wakes of innovations" (Boland et al. 2007) that traverse through the entire digital ecosystem.

Information Systems (IS) scholars have studied technology-enabled innovations in organizations (Sambamurthy et al. 2003; Swanson 1994). However, most of these studies are confined within the context of firms that are innovating following the logic of modularity. Particularly many studies focus on how technology can improve the efficiency and effectiveness of an organization with a fixed boundary (Yoo et al. 2010). As such, existing theories on innovation in IS literature cannot effectively capture the dynamic and relentless nature of innovations that we see in contemporary digital ecosystems (Tiwana et al. 2010).

In order to fill this void in the literature, we draw on models of generativity from network biology. A network biology perspective focuses on the interactions among existing components (i.e., genes or proteins in biology) as a way to understand the dynamic structure of cells and organisms (Fleming et al. 2001; Fleming et al. 2004; Kauffman 1993; Kitano 2002; Levinthal 1997). Specifically, the field of network biology (in the molecular sense) is concerned with how interactions among genes produce functionally differentiated cell types in an organism: such interaction patterns among genes are captured

through a co-expression network (Ravasz et al. 2002; Ruan et al. 2010; Stuart et al. 2003; Zhang et al. 2005). Using co-expression genetic networks, biologists cluster genes according their co-expression patterns. Such networks are effective in identifying and characterizing patterns of gene interactions, especially when the number of genes is large. In this paper, we draw from network biology to explore the underlying structure of generativity in an open digital ecosystem. In particular, we focus on how third-party developers combine diverse APIs to create new plug-ins . We use a co-expression network of APIs to explore clusters of APIs that are repeatedly used together to generate plug-ins as a way of understanding the structure of the generativity of an open digital ecosystem (Ravasz et al. 2002). Specifically, we can ask:

a) What is the underlying structure of generativity in an open digital ecosystem?
b) What are the patterns of control in design in an open digital ecosystem?

We answer these questions using the data set collected from WordPress.org, the world's largest blog service platform. The WordPress ecosystem is particularly known for its vibrant and diverse set of functions provided through a large number of plug-ins, most of which are designed and built by third-party developers. In what follows, we first review existing literature on modularity and generativity. Second, we introduce a network biology approach and the notion of co-expression networks and discuss how they can be used to explain generativity in an open digital ecosystem. Third, we present our empirical study. We conclude by discussing the theoretical and managerial implications of using the network biology perspective in understanding the generativity of a digital ecosystem.

Literature Review

Modularity

(Simon 1962) proposed the notion of modularity as a way of understanding a complex system via a hierarchical relationship between the system and its modules. One can use a modular architecture to represent a complex system divided into a set of subsystems that perform specific functions and their relationships (Ulrich 1995). As such, modularity offers simplicity in dealing with a complex system as one can focus on an overall architectural scheme that defines the boundary of the system, the modules that make up the system, and interfaces among the modules, while leaving the detailed design of the modules to others. This allows an effective division of labor among different actors during the design and production of a complex system (Sosa et al. 2004; Staudenmayer et al. 2005).

Not only does modularity reduce the complexity in design and production of a product, it also increases its flexibility by allowing the "mixing-and-matching" of various modules (Sanchez et al. 1996). Such a "mixing-and-matching" strategy is possible as one can replace one module with another as long as they both confirm to the same standardized interface. For example, with personal computers, a user can use any video board or a monitor as long as they follow the same standard and protocol. Firms can then increase product variations by leveraging modules in an increasingly "intermeshed" way (Sanchez 2004).

Traditionally, scholars have focused on the role of the firm which controls the dominant design of a product through its architectural knowledge (Clark 1985; Henderson et al. 1990). Recently, however, scholars have begun to propose that heterogeneous firms can autonomously coordinate (either explicitly or implicitly) their designs and productions of modules without the intervention of the architecture owner when the architecture is mature enough with a set of clearly articulated interfaces (Staudenmayer et al. 2005). In this sense, a product can be considered as a network of peer modules that share the same interfaces in order to function as a whole (Sosa et al. 2005). By leveraging such flexibility stemming from modularity, a firm which owns the architecture can dramatically increase variations in their product and the overall agility of their organization without incurring high transaction cost (Sambamurthy et al. 2003; Sosa et al. 2004).

While such a network perspective of modular innovations helps us understand production innovations, these models make implicit assumptions that the boundary of system is fixed and clearly articulated *a priori* by a central authority (Nickerson et al. 2004). Therefore, the flexibility of a product and its variations take place within a confinement of a given architectural scheme (Ulrich 1995). Here, modules are, no matter how diverse they are, *product-dependent*. That is, the product is designed first and then modules are designed (Gulati et al. 2012; Yoo et al. 2010)

However, increasingly, in open digital ecosystem, where modules are designed first without considering particular product architecture, modules are *product-agnostic*. For example, when Google introduced its popular Google Maps APIs, these API modules were not designed for specific products. Yet, in a short time, these Google Maps APIs were combined with multitude of other resources available on the Internet (e.g., Craigslist, Facebook, or private databases) to produce thousands of, what we in the vernacular call, mash-ups. Most of these mash-ups were not intended, designed, or approved by Google. Now these Google Maps APIs are integrated with different types of hardware such as mobile phone, portable navigations, digital cameras and automobiles, continuing to create new types of innovations that were not intended by Google. Here, there appears to be no explicit architecture that provides a fixed boundary of a system. In a way, innovations in digital ecosystems are unbounded (Yoo et al. 2010). In this paper, we offer a new theoretical perspective on innovations in digital ecosystem that do not require *a priori* architectural knowledge defining the boundary and interfaces of a system.

Generativity of a Digital Ecosystem

The idea of generativity is a useful concept to understand the dynamic and unbounded aspect of digital innovations (Yoo et al. 2010). Zittrain (2006) focuses on particular features of digital technology and the behavioral traits of actors in order to explain the generativity of digital artifacts. In this paper, we adopt the (re)combinatorial view of digital innovation, focusing on the recombination of existing software code modules ,or APIs, to create new features of an ecosystem as they are implemented as complementary products ,or plug-ins, developed by third-party developers (Chesbrough et al. 2006; Tiwana et al. 2010). Specifically, we define the generativity of a digital ecosystem as its reproductive capacity to produce unprompted and uncoordinated changes in its structure and behavior without the control of a central authority by utilizing existing software modules that can be recombined in novel ways. Digital ecosystem refers to a set of a platform, APIs, and complementary products (Cusumano et al. 2002; Tiwana et al. 2010) together with a set of actors including as platform owner and third party developers (Ghazawneh et al. 2012b). Therefore, generativity can occur in digital ecosystems where mutually interdependent resources are combined with each other by third party developers.

At the core of our conceptualization of generativity of a digital ecosystem is a rich set of heterogeneous APIs that produces an ever-changing landscape (Nickerson et al. 2004). Many successful digital ecosystems, such as Facebook, Twitter, and WordPress, critically depend on the innovations by heterogeneous third-party developers who create new digital artifacts (such as software plug-in programs or Apps) in pursuit of their own self-interests and ideas. However, these generative actions by third-party developers are not entirely random, as they often rely on existing APIs in building their own digital artifacts (Fleming et al. 2004; Nelson et al. 1982). APIs are developed not only from the platform providers such as Google, but also from firms that do not have their own platform services. Therefore, the generativity of a digital ecosystem is an outcome of purposive redefinition and re-combination of existing heterogeneous APIs (Chesbrough et al. 2006). Incredibly rich sets of digital artifacts (namely, plug-ins) are derived from the simultaneous interactions among pre-existing APIs. In an abstract way, one can argue that new breeds of digital artifacts in a digital ecosystem can be created through the creative recombination of existing modules. To explain the generativity of a digital ecosystem, we adopt a network biology perspective that describes how the similar sequences can produce many different cells that lead to diverse functions (Kauffman 1993).

Theory Development

A Network Biology Approach

One of the key goals of network biology is to understand the generativity of biological systems containing multiple cell types deriving from the same genetic code. These multiple cell types, exhibiting different functions, are produced through selective, non-linear, and non-additive interactions among genes (Kitano 2002). A co-expression network determines how different genes are co-expressed in phenotypes through developmental processes such as cell differentiation and morphogenesis. Interactions between genes/proteins, or epistasis, provide the structural backbone for networks. A gene is the basic functional unit in any genome whose information is stored as DNA. These DNA are eventually translated into amino

acids and linearized as a protein, the basic functional and structural unit of a cell. Thus, a co-expression network, via the highly dynamic and generative patterns of interactions among genes, plays an essential role in the process by which genotypes give rise to phenotypes, determining how different cells perform different functions within an organism although they all share the same pool of genes (Barabási et al. 2004; Ruan et al. 2010; Strogatz 2001; Stuart et al. 2003).

Although the pattern of interactions among genes is non-linear and selective, certain combinations of genes are repeatedly used across a diverse set of functions. These recurring patterns of gene interactions form the basis of clusters of functionally similar gene interactions, with varying degree of varieties . Using a co-expression network, one can capture this dynamic and non-linear pattern of interactions across clusters of gene interactions. Specifically, one can represent neighboring relationships among clusters of gene interactions based on the presence of modifier genes (Ruan et al. 2010).

Furthermore, the combinations of these clusters (i.e., recurring patterns of gene interactions) form the basis of different cell types. In order to capture the diversity of cells, therefore, we use the topology landscape of the co-expression network via a topological overlap matrix. A topology represents the structural characteristics of a network. A topological model presents a bottom-up approach in identifying clusters of gene interactions that are frequently used together (Albert et al. 2003). A topology, therefore, is useful in understanding the large structure of a complex network (Jeong et al. 2000). Using a topological model, we can explore the possible presence of recurring patterns of gene interactions in a cell (Zhao et al. 2010). In a topological overlap of a network, clusters of recurring gene interactions are interconnected with one another (Ravasz et al. 2002). By focusing on the topological overlap, therefore, we do not need to focus on individual genes, but the network of clusters of gene interactions to simplify the complex network.

Using topological overlaps, we can also construct a landscape of gene interactions (Kauffman 1993). The rugged shape of a landscape is determined by clusters of recurring gene interactions and the degree of variety among gene interactions in each cluster. These clusters of recurring gene interactions are scattered in the landscape based on their similarities. The interactions between these clusters create considerable diversity in the biological system. Therefore, increased rugged region scattered across a landscape can make the interaction of modules even more vibrant and dynamic, which increases the generativity of the biological system.

A Network Model of Generativity in Digital Ecosystems

An increasing volume of modern digital artifacts, such as plug-ins of web services or mobile apps, consist of existing software modules (like APIs) and additional software codes that glue these modules together and add additional functionalities such as user interface. Since designers create digital artifacts by combining existing software modules, we can use co-expression networks to characterize the way different modules are used to create these digital artifacts. Further, as certain combinations of APIs are likely to be used more frequently than others, we can identify clusters of APIs in the same way biology scholars identify clusters of recurring gene interactions. The pattern by which these clusters of APIs are used for different plug-in software products in an open digital ecosystem forms its generative structure. Furthermore, this structure can be simplified by identifying topological overlaps of clusters.

As an example, in Figure 1, we have five different APIs (A0, A1, A2, B0, and B1). APIs labeled as A are developed by one firm and APIs labeled as B are developed by another firm. These five APIs are used to produce two different types of plug-ins, represented by two rectangular boxes. Type 1 plug-ins (with a solid line) use a cluster of APIs including A0, A1, B0 and B1; Type 2 plug-ins (with a dotted line) use a cluster of APIs including A0, A1, A2, and B0. From this, we can identify a topological overlap between the two plug-in types that shares A0, A1 and B0.

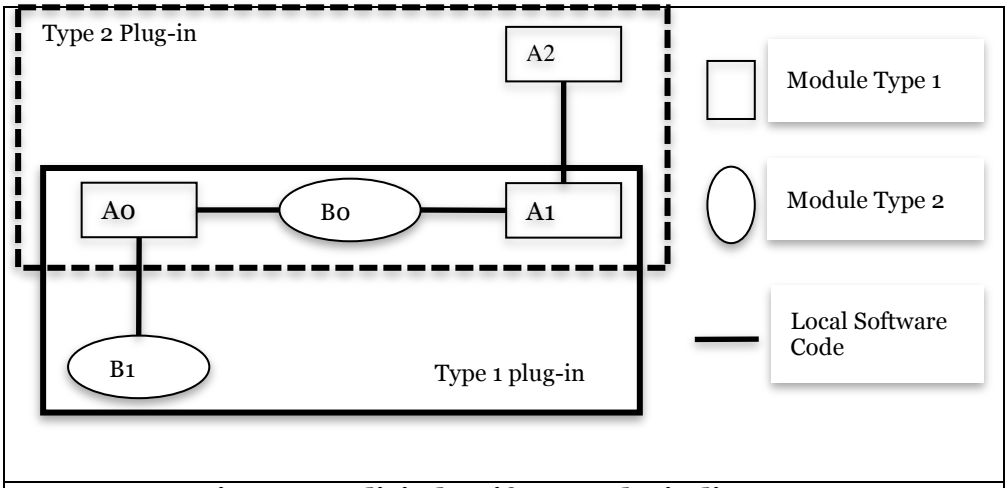

Figure 1. A digital artifact topologic diagram

An important characteristic of an open digital ecosystem is the absence of a central design authority who coordinates and integrates the innovation activities of others (Tiwana et al. 2010). Using the topological overlap and the co-expression network of APIs, we can examine how different plug-ins are created through combinations of different APIs. Furthermore, we can explore the landscape of WordPress ecosystem by investigating the way in which certain APIs are repeatedly used to produce different types of plug-ins with an increasing degree of varieties. The relationship among API clusters as shown in topological overlaps can be understood as the underlying generative structure of the entire ecosystem. We can further explore the nature of control of an ecosystem by examining the ownership of individual APIs that belong to different plug-in types. Specifically, in a tightly controlled ecosystem, the focal firm, which owns the platform would control most of the key APIs that are frequently used by plug-ins. In contrast, in an open ecosystem, the ownerships of APIs in these clusters are expected to distribute more widely. In this paper, we empirically explore two aspects of the generativity of an open digital ecosystem – its underlying structure and the nature of the control of the ecosystem. Ultimately, we offer a new theoretical and empirical framework of generativity of an open digital ecosystem, drawing on a network biology perspective.

An Empirical Study

Data

We collect data from WordPress.org (WordPress from hereon), which is the world's leading blog platform. Wordpress is an excellent example of how generativity occurs through the participation of third-party developers in an open digital ecosystem. It has over 22,000 plug-ins as of December 2012. Most of these plug-ins were developed by third-party developers by connecting external web services APIs such as Google, Facebook and Twitter together with a limited set of internal functions that WordPress provides through its own set of APIs. In this paper, we refer to non-WordPress APIs such as Google, Facebook, and Twitter as 'external APIs'. The interactions between the original WordPress APIs and external API modules form the "genetic basis" of an incredibly rich array of plug-ins. Each plug-in used APIs 29 times on average showing 82 standard deviations (with the minimum number of APIs used being 0 and the maximum number 4796). We collected the latest version of available 13,491 plug-ins written in PHP code from Wordpress that were available as of December 2012[2].

[2] Although there are more than 22,000 plug-ins, only 13, 491 plug-ins are written PHP code that allow us to read the source code.

Analytical Approach

We use a network biology perspective to analyze the interaction among various APIs used to create WordPress plug-ins, as follows: Firstly, we downloaded the source code for all plug-ins and used a text-mining program written in JavaScript to identify the APIs that are used in each plug-in. Specifically, we identified APIs used in plug-ins in two different ways. We identified 93 WordPress APIs from the WordPress webpage and 915 external APIs from www.programmableweb.com. Out of 915 available external APIs, we found that only 195 of them actually were used in our sample of 13,491 plug-ins. Through our text-mining script, we also identified how many times each API was used for each plug-in. From this, we constructed a 13,491 plug-ins × 288 APIs (93 internal APIs + 195 external APIs) matrix to capture the interaction of APIs among plug-ins in the WordPress ecosystem. We use this data set to answer our two research questions using different analytical approaches.

In order to answer the first question by exploring the underlying structure, we built a plug-ins co-expression network from our original data set (Zhang et al. 2005). Each node represents a plug-in and each edge is the connection strength of each pair of plug-ins across APIs. The weight of the edge is greater when a pair of plug-ins use the same APIs more frequently. Using this co-expression network, we then built on an adjacency matrix of plug-ins, which represent the connection strength between plug-ins, calculated by Pearson correlation. An adjacency matrix is measured between plug-ins i and j represented as a_{ij} so that we generate a 13,491 × 13,491 matrix. For example, a_{ij} = 1 when plug-ins i and j share the same set of APIs. If a plug-ins does not use the same APIs, a_{ij} will be 0. As we used the data set representing actual number of API usage, we can get the weighted measures. The adjacency matrix is the foundation to get the topological overlap. Topological overlap measures the relative inter-connectedness (Ravasz et al. 2002). It represents the distance between plug-ins based on the adjacency matrix. The number of direct neighbors of a plug-in i is $k_i = \sum_j a_{ij}$. Each k_i shows how a plug-in is similar with its neighboring plug-ins in terms of APIs used. The number of APIs shared by plug-in i and j is $\sum_{u \neq i,j} a_{iu}a_{ju}$. From this process, we can get the topological overlap as the following. The value will be 0 if plug-in i and j are not connected and both of them do not share any APIs.

$$Topological\ overlap\ measure = \frac{\sum_{u \neq i,j} a_{iu}a_{ju} + a_{ij}}{\min(k_i + k_j) + 1 - a_{ij}}$$

We can dichotomize clusters of plug-ins from the similar topological overlap measure. From this process, topologically highly overlapped plug-ins are likely to form clusters in a network. Clusters can be classified differently depending on the topological overlap within a cluster. The main advantage is to have robustness in understanding complex network in a simplified way (Yip et al. 2007). An hierarchical clustering tree, or dendrogram, can be drawn from the dichotomized clusters. Each leaf at the terminal end of a branch indicates a plug-in. The height is close to 1 if a cluster of plug-ins has a high topological overlap measure, and it will be close to 0 if the topological overlap measure is low. The height of dendrogram implies the different degree of topological overlap. Using the height, we can identify which plug-ins are closely related to each other. From this process, we can also identify the clusters of plug-ins connected with one another based on shared APIs. From the dendrogram, we can draw a topological overlap plot that provides a simplified network from clusters of plug-ins. The row and columns are characterized from the dendrogram so that we can easily identify how plug-ins are connected with one another through shared APIs even with plug-ins posited in different types of branches. Red indicates the higher value of the topological overlap measure, while yellow is the low value of the topological overlap measure. Thus, using the topological overlap plot, we can begin to understand the recurring patterns of APIs used for plug-ins.

To answer our second question concerning the control of an open digital ecosystem, we built another network of 288 x 288 APIs to specifically address how individual APIs are connected with each other. Thus, in this network, each node is an API, and each edge represents two APIs are shared by plug-ins. Through this network, we are focusing on the centrality of APIs. So, unlike topological overlap analysis, we used unweighted network using 0 and 1. A 0 implies that a pair of APIs is not used together by plug-ins, while 1 indicates that a pair of APIs is used together by plug-ins. We analyze the distribution of APIs rather than the network of APIs to understand exactly which APIs are essential in creating plug-ins. The degree of an API network represents the number of APIs that the focal API connects to (used by the same

plug-in). Thus, understanding the distribution lets us specifically know the extent of API usages across plug-ins.

Results

Underlying Structure of Generativity in Open Digital Ecosystems

Based on the topological overlap measure, we used R to draw a topological overlap plot as shown in Figure 2, where the different overlapped clusters are color-coded to represent the extent of overlap. In the topological overlap plot, each row and column corresponds to the same dendrogram so that we can understand how each plug-in is connected with one another.[3] By matching the plug-ins under categorized branches, we can understand the similarities among plug-ins across the scattered overlapped region.

We find tetragonal regions composed of different colors (i.e., red and yellow) in Figure 2. The tetragonal region, represented as a heat map, illustrates that the interactions of particular plug-ins mainly occurred within the boundary of the region rather than other regions. The tetragonal regions correspond to interactions of each plug-in. Different colors within a tetragonal region indicate different degrees of topological overlap. Yellow and red colored regions mean different degrees of topologically overlapped parts. The red color indicates the densely overlapped part within a topology, while the yellow color indicates the sparse part. Therefore, different colors in Figure 2 enable us to identify the degree of interaction among clusters of plug-ins. This allows us to see which clusters are more vibrant than others. The size of the tetragonal regions refers to the main boundary of the occurrence of interactions among clusters. From this graph, we can describe how many particular plug-ins share the same APIs from the size of the tetragonal region, as the length of branches relates to the degree of variants of a cluster of plug-ins. New plug-ins appear to continuously emerge from the original one. In these cases, the length of branches is often extended, implying that many new changes have occurred. A short branch indicates that the particular cluster is not popular to third party developers.

[3] Since Figure 2 contains 14,391 analyzed plug-ins, the name of each plug-in could not be specifically noted on the figure.

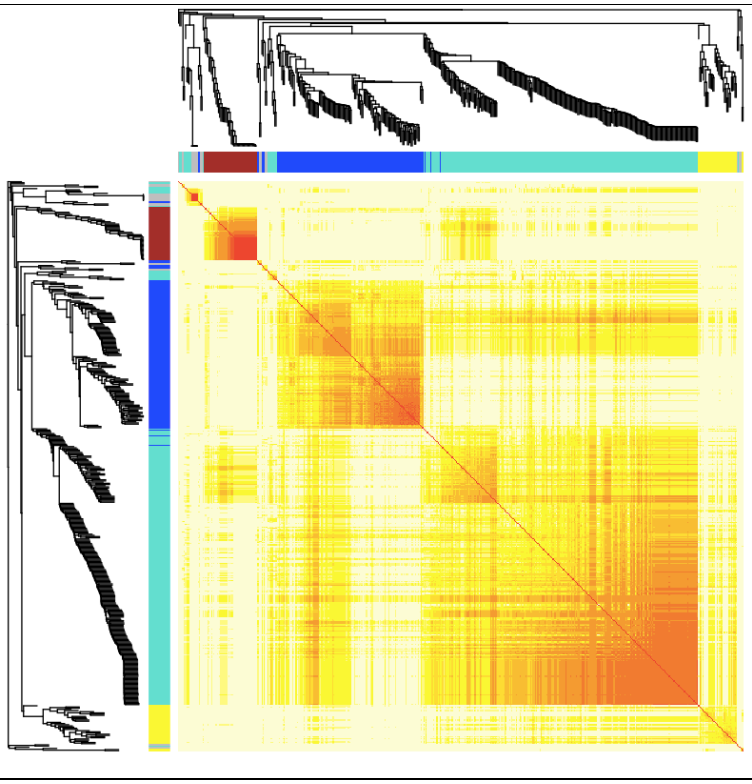

Figure 2. A topology overlap plot of 13,491 plug-ins

For example, in Figure 2, there are three large different squares along the diagonal line represented in different colors on each row and column. The different squares imply that they do not share the same API with one another. The different colors in each square indicate that some APIs in the cluster are more connected with other APIs thus producing more variants of plug-ins than other APIs. For example, there is a discernible yellow colored rectangle in the 1st quadrant in Figure 2. Unlike the squares, the rectangle implies that there are more plug-ins in the region that share a common set of APIs than other regions. The topological plot let us see which combinations of APIs are repeatedly used forming an underlying genetic architecture for popular clusters of plug-ins.

Our analysis of a co-expression network provides an answer to the question of generative structure of an open digital ecosystem. The hierarchical relationship of APIs, plug-ins, and clusters-of plug-ins as shown in Figure 2 indicates that certain combinations of APIs form the *genetic foundation* of highly vibrant region of plug-ins that attract large-number of third-party developers. These developers bring their own "modifier" APIs to combine with the APIs that form the genetic basis of the cluster. Furthermore, our analysis shows that there are relatively few genetic structures that form highly volatile clusters of plug-ins. In order gain further insight on the nature of the generativity in an open digital ecosystem, we explore who controls these genetic foundations by exploring API networks.

Control of an Open Digital Ecosystem

To answer the second research question, we constructed an undirected network where nodes are APIs and edges are API-API relationships based on the 13,491 plug-ins × 288 APIs matrix. There exist 288 nodes and 7516 edges in the API module network. We used R to calculate the distribution of APIs. Figure 3 shows a visual representation of this distribution.

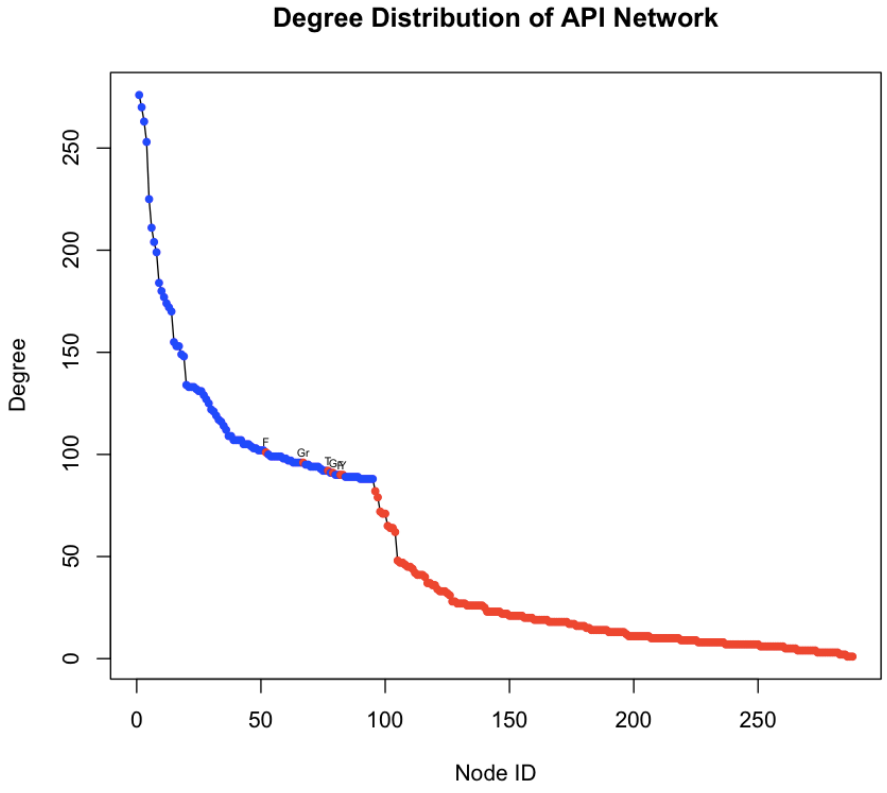

Figure 3. Color Coded API Distribution in Word Press Ecosystem

When we visually inspect the observed distribution of APIs, it seems to have a distinctive distribution pattern that is quite different from two well-known single distributions, a random distribution and a Power Law distribution. In order to confirm our visual inspection, we conducted the goodness of fit test to compare the degree distribution of our API network against two distributions that network nodes would most likely to follow.

First, we tested whether the API network is a random network. A random network assumes that all edges in the network are generated at the same rate so that repetitive edges between nodes do not exist (Erdős et al. 1960a; Erdős et al. 1960b). The degree of any node k has a probability distribution as follows:

$$p_k = \binom{n}{k} p^k (1-p)^{n-k},$$

where n is the number of nodes, m is the number of edges in a network, and $p = 2m/n(n-1)$ for undirected network. The test calculates the difference between the degree distribution of the observed network and the hypothesized random network. The comparison is possible from the Kolmogorove-Smirnov (KS) statistic (Press et al. 1992). The model is defined as the following.

$$D = \max|P^*(k) - P(k)|,$$

where $P^*(k)$ is the CDF of the data for the degrees of observed API network , and $P(k)$ is the CDF for the degrees of hypothesized random network. In KS statistic, D is the maximum distance between the CDFs of the observerd API network and of the hypothesized network. The model of goodness of fit test generates a

p-value that quantifies the probability if observed API network fits with a random network. Our results of the goodness of fit test show the p-value that is less than 0.01. Thus, the observed distribution does not fit to random distribution.

Second, we tested whether the network follows a Power Law distribution, which is the distribution most real world networks follow. With a Power Law distribution, any node k has a probability distribution as follows:

$$P(k) \approx k^{-\gamma},$$

where $P(k)$ is the degree distribution, and γ is a constant value around 2.2. To see if the observed API network follows a Power Law distribution or not, we tested a log-likelihood ratio test (Clauset et al. 2009). The basic idea behind the log-likelihood ratio test is computing the likelihood of the API network under two distributions. The first distribution is from the API network, and the second distribution is from a power law distribution. Likelihood ratio is the ratio of these two likelihoods as follows (Clauset et al. 2009).

$$\mathcal{R} = \sum_{i=1}^{n} [\ln p_1(x_i) - \ln p_2(x_i)],$$

where $\ln p_1(x_i)$ can be thought of as the log-likelihood from the distribution of API network, and $\ln p_2(x_i)$ can be understood as the log-likelihood from the distribution of random network. Our results show that p-value is less than 0.01, thus we can conclude that the observed distribution does not fit to Power Law distribution. Therefore, we can conclude that although there is a particular structural pattern as the way API modules are co-expressed in plug-ins, the distribution does not follow either a random or a power law distribution.

To further explore the nature of the distribution of APIs, we analyzed how each WordPress API and external API is posited on the distribution. In Figure 3, a blue color indicates a WordPress API, while a red color indicates an external API. From the different color-coded analysis, we can clearly see that two distinctively different distributions are combined to form a hybrid distribution.

We can make two important observations. First, APIs developed by WordPress and those developed by other firms form two distinctively different distributions. Furthermore, the WordPress APIs occupy the head region of the overall hybrid distribution, while the external APIs occupy the tail region. We speculate that the first region represents the APIs that form the genetic foundations of plug-in clusters. The second region, on the other hand, we believe, are the "modifier" gene APIs that produces variants through epistatic effects. Given the differences in the frequency of usages, it is clear that WordPress APIs provide the genetic basis of most plug-ins. This suggests that even though WordPress.org is an open platform, WordPress exercises an extensive control of the evolution and generativity of the ecosystem through its own APIs. What is remarkable is that it does so only through 93 APIs.

Second, what is also interesting is that there are few external APIs that were found in the region of mostly WordPress APIs. To further explore this, we identified the owners of those APIs. In Figure 3, we marked those external APIs found in the first region with letters to indicate the firm who own them. For example, F is an API provided by Facebook, Gr is an API provided by Gravatar, G is an API from Google, T is an API provided by Twitter, and finally Y is an API from Youtube. This suggests that these five external APIs are part of genetic foundation of popular plug-ins in the WordPress ecosystem. Our results suggest that not all external APIs serve only as modifier gene. Some external APIs become an integral part of the genetic foundation of the WordPress ecosystem.

Discussion

In this study, we ask whether there is an underlying generative structure in an open digital ecosystem even though there is no particular central authority. In addition, we explore whether there is a proper method to effectively detect the structure of a complex network. Our results complement previous studies on the system design and architecture (Baldwin and Clark, 2000) by showing that there can be an architecture of generativity without a central authority. Contrary to the prior work based on modularity as the basis of complex system design, our work shows that incredibly complex ecosystem emerge in an orderly way without having a central design.

A network biology perspective offers a powerful lens to understand the ever-changing landscape of a digital ecosystem. Kauffman (Kauffman 1993) characterizes the rugged regions such as different clusters of plug-ins in Figure 2. The topological overlapped APIs are understood as local optimums that are defined as "basins of attractors" that can lead to the generation of many new plug-ins (Levinthal 1997). The basins of attractors decide the degree of development of a particular functional class. Therefore, highly topologically overlapped areas in the dendrogram can be understood as a basin of attractor in a network biology perspective, as it is likely that we will find the emergence of new plug-ins in that region. The combinations of APIs that are used in the basins of attractor form the genetic foundations of the clusters of plug-ins. Such genetic foundations of basins of attractors then provide a basic set of APIs that third-party developers can use to experiment with new APIs and new functions that lead to the emergence of new types of plug-ins in the ecosystem.

Our study also provides a new perspective on the relationship between platform owner and third party developers in terms of generativity. Innovation mechanism in a digital ecosystem is not limited to the functional enhancement of existing platforms, but also includes the emergence of new clusters of plug-ins that derive from the interactions between platform owners and third-party developers. Our study shows that platform owners must shift their attention to the control of the genetic architecture that form the basins of attractors in their ecosystem. The ownership of a platform becomes irrelevant if other firms control those genetic foundations.

At the same time, our results show that the platform owner needs to relent and let some of the popular APIs become a part of a genetic foundation. In the case of WordPress, it has allowed five different external APIs that are frequently used together with other WordPress APIs. Such an insight might provide a theoretical basis of what happened in the competitive landscape of mobile ecosystem represented by Apple, Google, Blackberry and Windows.

This paper contributes to our understanding of generativity and innovation with relation to each other by focusing on the base structures of digital artifacts. Underlying mechanisms of innovation can be understood in two different ways. The analysis of topologic overlap informs us about the interactions among plug-ins by showing generic structure of APIs. Such a new approach requires the integration of the network aspects based on the existing notion on innovation. Such integration is derived from the emergence of a digital ecosystem that leads organizations to freely intermingle modules showing different interfaces without barriers. To understand the interaction of modules in a digital ecosystem, this paper seeks to understand market dynamism in a novel way and suggests a strategic response in a network biology perspective.

There are a number of limitations in this study. Although each plug-in shows functionally different traits, we cannot normalize them in a systematic way. This limitation also makes it hard to find more dynamic innovation patterns over time. To overcome the limitations, we will collect time-series plug-in software code data across multiple years to understand the dynamics of innovation.

Overall, this research provides a useful guide to help us understand innovation in a digital ecosystem in a more systematic way. By unearthing the architectural structure of generative innovation in a digital ecosystem, this study provides a strategic vantage view to understand how continuous innovation can be pursued in an organization in a dynamic environment. This model can provide a clue on how individuals in organizations have to do away with their ordinary routine work for progress and how managers need to arrange the individual changes by proposing new directions of innovation.

Conclusion

An open digital ecosystem represents the most dynamic field of innovation. This dynamism derives from unbounded systemic features causing generativity to emerge in unpredictable ways. However, within current innovation discourse, it is difficult to fully explain the phenomenon of an open digital ecosystem as current studies of innovation are based on the assumptions of bounded systems with the control of a central authority (i.e., hierarchical architecture). To complement existing studies, we begin to draw on concepts from network biology. A co-expression network provides an ideal way of explaining the mechanism of generativity. In particular, comparing the topology of modules is a reliable method as it focuses on mutually interacting modules. If we can specifically find the pattern, we can make a predictable model for the generativity in a digital ecosystem. While we acknowledge the limitation in generalizing the case of the WordPress ecosystem, we hope that this study will be an initial step to explore both theoretical and methodological applications in understanding new system order. We also want to understand how the generative structure changes over time as new APIs emerge. Such dynamic analysis will also help us answer how platform owners should exercise its control so that the ecosystem system can continue to grow, while it does not lose control over its genetic foundations.

References

Albert, R., and Othmer, H. G. "The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster," *Journal of theoretical biology* (223:1) 2003, pp 1-18.

Arthur, W. B. *The nature of technology: What it is and how it evolves* Free Press, New York, 2009.

Baldwin, C. Y., and Clark, K. B. "Design rules, Vol. 1: The power of modularity,") 2000.

Barabási, A. L., and Oltvai, Z. N. "Network biology: understanding the cell's functional organization," *Nature Reviews Genetics* (5:2) 2004, pp 101-113.

Boland, R. J., Lyytinen, K., and Yoo, Y. "Wakes of innovation in project networks: The case of digital 3-D representations in architecture, engineering and construction," *Organization Science* (18:4) 2007, pp 631-647.

Chesbrough, H. W., Vanhaverbeke, W., and West, J. *Open innovation: Researching a new paradigm* Oxford University Press, USA, 2006.

Clark, K. B. "The interaction of design hierarchies and market concepts in technological evolution," *Research Policy* (14:5) 1985, pp 235-251.

Clauset, A., Shalizi, C. R., and Newman, M. E. J. "Power-law distributions in empirical data," *SIAM review* (51:4) 2009, pp 661-703.

Cusumano, M. A., and Gawer, A. "The elements of platform leadership," *MIT Sloan management review* (43:3) 2002, pp 51-58.

Erdős, P., and Rényi, A. "On the evolution of random graphs," Citeseer, 1960a.

Erdős, P., and Rényi, A. "On the evolution of random graphs," *Magyar Tud. Akad. Mat. Kutató Int. Közl* (5) 1960b, pp 17-61.

Fleming, L., and Sorenson, O. "Technology as a complex adaptive system: evidence from patent data," *Research Policy* (30:7) 2001, pp 1019-1039.

Fleming, L., and Sorenson, O. "Science as a map in technological search," *Strategic Management Journal* (25:8–9) 2004, pp 909-928.

Garud, R., and Kumaraswamy, A. "Changing competitive dynamics in network industries: An exploration of Sun Microsystems' open systems strategy," *Strategic Management Journal* (14:5) 1993, pp 351-369.

Ghazawneh, A., and Henfridsson, O. "Balancing platform control and external contribution in third-party development: The boundary resources model," *Information Systems Journal* (22:2) 2012a.

Ghazawneh, A., and Henfridsson, O. "Balancing platform control and external contribution in third–party development: the boundary resources model," *Information Systems Journal*) 2012b.

Gulati, R., Puranam, P., and Tushman, M. "Meta–organization design: Rethinking design in interorganizational and community contexts," *Strategic Management Journal* (33:6) 2012, pp 571-586.

Henderson, R. M., and Clark, K. B. "Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms," *Administrative Science Quarterly*) 1990, pp 9-30.

Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., and Barabási, A. L. "The large-scale organization of metabolic networks," *Nature* (407:6804) 2000, pp 651-654.

Kauffman, S. A. *The origins of order* Oxford University Press New York, 1993.

Kitano, H. "Systems biology: a brief overview," *Science* (295:5560) 2002, pp 1662-1664.

Levinthal, D. A. "Adaptation on rugged landscapes," *Management science* (43:7) 1997, pp 934-950.

Nelson, R. R., and Winter, S. G. *An evolutionary theory of economic change* Belknap press, 1982.

Nickerson, J. A., and Zenger, T. R. "A knowledge-based theory of the firm—The problem-solving perspective," *Organization Science* (15:6) 2004, pp 617-632.

Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N., and Barabási, A. L. "Hierarchical organization of modularity in metabolic networks," *Science* (297:5586) 2002, pp 1551-1555.

Ruan, J., Dean, A. K., and Zhang, W. "A general co-expression network-based approach to gene expression analysis: comparison and applications," *BMC systems biology* (4:1) 2010, p 8.

Sambamurthy, V., Bharadwaj, A., and Grover, V. "Shaping agility through digital options: Reconceptualizing the role of information technology in contemporary firms," *Mis Quarterly*) 2003, pp 237-263.

Sanchez, R. "Creating modular platforms for strategic flexibility," *Design Management Review* (15:1) 2004, pp 58-67.

Sanchez, R., and Mahoney, J. T. "Modularity, Flexibility, and Knowledge Management in Product and Organization Design," *Strategic Management Journal*) 1996, pp 63-76.

Simon, H. A. "The architecture of complexity," *Proceedings of the American philosophical society* (106:6) 1962, pp 467-482.

Sosa, M. E., Agrawal, A., Eppinger, S. D., and Rowles, C. M. "A network approach to define modularity of product components," 2005.

Sosa, M. E., Eppinger, S. D., and Rowles, C. M. "The misalignment of product architecture and organizational structure in complex product development," *Management science*) 2004, pp 1674-1689.

Staudenmayer, N., Tripsas, M., and Tucci, C. L. "Interfirm Modularity and Its Implications for Product Development*," *Journal of Product Innovation Management* (22:4) 2005, pp 303-321.

Strogatz, S. H. "Exploring complex networks," *Nature* (410:6825) 2001, pp 268-276.

Stuart, J. M., Segal, E., Koller, D., and Kim, S. K. "A gene-coexpression network for global discovery of conserved genetic modules," *Science* (302:5643) 2003, p 249.

Swanson, E. B. "Information systems innovation among organizations," *Management science*) 1994, pp 1069-1092.

Tiwana, A., Konsynski, B., and Bush, A. A. "Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics," *Information Systems Research* (21:4) 2010, pp 675-687.

Ulrich, K. "The role of product architecture in the manufacturing firm," *Research Policy* (24:3) 1995, pp 419-440.

Yip, A. M., and Horvath, S. "Gene network interconnectedness and the generalized topological overlap measure," *BMC bioinformatics* (8:1) 2007, p 22.

Yoo, Y., Boland, R. J., Lyytinen, K., and Majchrzak, A. "Organizing for Innovation in the Digitized World," *Organization Science* (23:5) 2012, pp 1398-1408.

Yoo, Y., Henfridsson, O., and Lyytinen, K. "Research Commentary---The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research," *Information Systems Research* (21:4) 2010, pp 724-735.

Zhang, B., and Horvath, S. "A general framework for weighted gene co-expression network analysis," *Statistical applications in genetics and molecular biology* (4:1) 2005, p 1128.

Zhao, W., Langfelder, P., Fuller, T., Dong, J., Li, A., and Hovarth, S. "Weighted gene coexpression network analysis: state of the art," *Journal of Biopharmaceutical Statistics* (20:2) 2010, pp 281-300.

Zittrain, J. "The generative internet," *Harvard Law Review* (119) 2006, pp 1974-2040.