Growth, Complexity, and Generativity of Digital Platforms: The Case of Otto.de

Completed Research Paper

Daniel Fürstenau
Freie Universität Berlin
School of Business & Economics
Garystr. 21, 14195 Berlin, Germany
daniel.fuerstenau@fu-berlin.de

Hannes Rothe
Freie Universität Berlin
School of Business & Economics
Garystr. 21, 14195 Berlin, Germany
hannes.rothe@fu-berlin.de

Abayomi Baiyere
Copenhagen Business School
Department of Digitalization
Howitzvej 60, 2000 Frederiksberg, Denmark
aba.digi@cbs.dk

Matthias Schulte-Althoff
Freie Universität Berlin
School of Business & Economics
Garystr. 21, 14195 Berlin, Germany
matthias.schulte-althoff@fu-berlin.de

Dieter Masak
Plenum AG
Am Flughafen
60549 Frankfurt am Main, Germany
dieter.masak@plenum.de

Kai Schewina
Freie Universität Berlin
School of Business & Economics
Garystr. 21, 14195 Berlin, Germany
kai.schewina@fu-berlin.de

Daria Anisimova
Freie Universität Berlin
School of Business & Economics
Garystr. 21, 14195 Berlin, Germany
daria.anisimova@fu-berlin.de

Abstract

We study the growth, complexity, and generativity of a digital platform—Otto.de. Through a longitudinal analysis of the company's 65 GitHub open source repositories over a period of 6.5 years, we find a) support for a superlinear growth pattern, b) a structural split in the platform into two clusters, and c) indication that more active repositories will be more generative and in turn more popular. These findings have implications for scholars studying platform evolution as well as for companies opening up their internal IT platforms.

Keywords: digital platforms, growth, complexity, generativity, platform evolution

Introduction

The literature on digital platforms has noted important issues of architectural design determining their long-term "evolvability" and success (Agarwal and Tiwana 2015; de Reuver et al. 2017; Tiwana et al. 2010). In this paper, we draw on important contributions highlighting the modularity and generativity of digital platforms. Generativity allows for unbounded new functions and services to emerge on top of a digital platform (Lyytinen et al. 2017). In accordance with Zittrain (2006, p. 1980), we refer to generativity as "a technology's overall capacity to produce unprompted change driven by large, varied, and uncoordinated audiences." It describes a platform's ability to generate new outputs, structures, or behaviors beyond the creators' original intentions (Henfridsson and Bygstad 2013; de Reuver et al. 2017; Zittrain 2008). Therefore, generativity is an important impetus and mechanism of innovation. However, with an exponentially growing user base, expanding heterogeneity and number of functions enabled by a dynamic and unexpected ecosystem, generativity in growing platforms come at the expense of complexity (Hanseth and Lyytinen 2010; Yoo et al. 2010).

Digital platforms reduce complexity with a modular architectural design (Baldwin and Clark 2000; Gawer 2014). A modular designs breaks complex systems by introducing a set of independent modules that are connected via well-defined interfaces and ensure the platform's core functionalities (Tiwana et al. 2010). This enables combinatorial innovation because peripheral modules can be changed without altering the platform's core (Yoo et al. 2010).

However, while previous contributions (e.g., de Reuver et al. 2017; Staykova and Damsgaard 2017; Tiwana 2013; Tiwana et al. 2010) have expanded our understanding of platform evolution, we currently lack a detailed understanding of generativity in the context of companies transforming their internal IT platforms towards an open platform model and evolving it continuously. Generative and open platforms promise access to unlimited resources, services, and actors (Hanseth and Lyytinen 2010), at exponential growth rates (Huang et al. 2017). However, this comes at the expense of added complexity which needs to be handled by modular design. Understanding the relationships between generativity, complexity, and growth is relevant and timely as both managers and scholars have realized the value of leveraging the wealth of talent and resources outside an organization's boundary (von Hippel and von Krogh 2003; Constantinides et al. 2018; Parker et al. 2017). However, empirical research on the relationship between these constructs are scarce. We are heralding an era where open platform models are increasingly providing avenues for organizations to generate and capture more value than would have been possible with the traditional closed platform models (Parker et al. 2017). Furthermore, it is not yet entirely clear how it is possible for an originally internal platform project to mobilize external developers for platform and business development on a continuous basis (von Hippel and von Krogh 2003; Constantinides et al. 2018; Parker et al. 2017).

The purpose of this paper is to *give insight into the growth, complexity, and generativity of a digital platform – Otto.de*. We give a detailed architectural overview of the platform's main characteristics and development since its inception in 2012[1]. This is important because many companies consider opening their internal IT platforms to external participants in the course of digital transformation. However, so far only a few companies have succeeded in completely digitalizing an existing business model. Our case company–Otto–is a noteworthy example that has successfully made such a digital transformation by shifting its business from catalogue shopping to e-commerce. Today, Otto is the second largest e-commerce company in Germany and its platform handles more than 90% of the company's order volume (€2.7 billion; 2 million site visits per day; up to ten orders per second). The platform is essential for Otto's vitality in a market in which many companies have disappeared. Moreover, the platform is interesting as our study context, as it adopts the open platform model and has evolved completely from a business unit's initiative to the most important source of value creation and appropriation for the company.

Analyzing the platform's growth via its 65 GitHub repositories, our data indicates a superlinear (quadratic) growth pattern. This pattern is fueled by two events: a restructuring of the platform via microservices and a reengineering to fit the demands of professional sellers. Furthermore, we find a structural split of the platform into two main clusters, giving evidence that time may reinforce structural separations in a

[1] An earlier version of the case analysis was presented as a short paper in Fürstenau et al. (2019). The analysis here is more advanced and uses a newly collected additional dataset.

platform. Finally, we find that increased activity on a repository is related to increased generativity and that increased generativity is associated with increased popularity. These findings hold implications for the literature on platform evolution and provide empirical evidence for growth and generativity dynamics of a digital platform.

Digital Platforms as Architectures That Facilitate Generativity

Platform theory is at the crossroads between disciplines (Baldwin and Woodward 2009; Gawer 2014). Platforms may be considered from an economic viewpoint as multi-sided structures supporting market exchanges ("match-making") (Evans and Schmalensee 2016; Rochet and Tirole 2003), which is not our focus here. Our view on platforms in this paper is motivated by another understanding of platforms. Platforms may also be considered from an architectural viewpoint as *software-based structures consisting of a relatively stable core and a dynamic periphery* (Tiwana et al. 2010). The distributed architecture of platforms enables rapid scaling and makes innovation in an unprecedented speed possible, which is our focal point of interest.

The cornerstone of digital platforms from an architectural viewpoint are autonomous software modules, which are coupled loosely to "mix-and-match" components without having to consider too strict dependencies between modules (Tiwana 2013). These modules can be developed by separate developer teams, allowing for a divide-and-conquer approach as well as a separation of concerns. Teams may be corporate-internal teams or volunteers, or a mix of both. In mixed corporate-volunteer settings, it may be possible to restrict access to some modules, while allowing access to others. An example is the GitHub platform, which allows public or private access to code repositories, depending on the status and affiliation of the contributor. By doing so, corporate platform development projects can maintain control over some parts of the code base, while profiting from external contributions in others. In the sense of platform openness, granting or restricting access to the code base thus represents an important boundary resource (Ghazawneh and Henfridsson 2013).

In contrast to closed software systems, opening a digital platform to outside contributors moves software development in corporate settings toward an open source software (OSS) model. Choosing the right degree of openness is thereby one of the most important and most difficult decisions for a platform company (Constantinides et al. 2018; Parker et al. 2016). On one hand, a high degree of openness attracts external developers who contribute to the platform and potentially spur innovation. On the other hand, externalization limits possibilities for control over the platform's growth and a loss of ownership (Tilson et al. 2010). Yoo et al. (2012, p. 1400) hold that "organizations must be designed to manage the delicate balance between generativity and control in the platform." Especially with the layered, modular architecture of digital platforms, it is important to achieve both goals–i.e., "stable and evolvable" (Constantinides et al. 2018; Wareham et al. 2014), to generate a competitive edge. Therefore, platform owners face the problem of making the design rules stable and flexible at the same time to limit the developers while giving them enough freedom (Tiwana et al. 2010). This paradoxical relationship between mutability and control is the platform company's central governance challenge (de Reuver et al. 2017; Tiwana et al. 2010). Tiwana et al. (2010) define three perspectives on design rules of a platform: the division of decision-making rights; control; and solo ownership versus joint ownership. Important strategic resources for the platform company are so-called boundary resources such as APIs or SDKs (Yoo 2013). APIs are control points facilitating data exchange and access to data or services (Evans and Basole 2016). They enable developers to "retrieve" data from relevant sources and use it for their own digital applications and services (ibid.).

The trend towards open source software is encouraged by the desire to combine in-house developments with external contributions within a platform model. OSS projects represent an innovative form of knowledge generation enabled by global digital infrastructures such as GitHub (Faraj et al. 2011). GitHub offers developers numerous features: collaborative working, tracking and tracing of the activity and development of the project, social networking, and several others (Wu et al. 2014). It is characterized by the development and modification of complex artifacts, based on a publicly available accessible code base (Borges and Tulio Valente 2018). Some public code is used to solve local problems and enable new projects (Zhang et al. 2014). Once developed, open source code may be reused in several internal as well as external projects (Zhang et al. 2014). This leads to the creation of large communities in which actors show common interest in a digital artifact (Eck and Uebernickel 2016). Actors involved in the project exchange knowledge with each other and collaborate on ideas, resulting in a high degree of generativity.

Hypotheses on Growth, Complexity, and Generativity

Growth of Digital Platforms

Researchers have long studied information systems' growth and found different growth patterns. One example is a study by Lehman and his colleagues who investigated the growth of five different software systems over time, such as the IBM operating system or the Lucent telecommunications system (Lehman 1980; Scacchi 2004). These researchers found that the growth of a software system over several releases approaches (in finite time) a (pseudo-)linear or inverse square model (Keßler 2013; Scacchi 2004). Lehman et al. (1997) conclude that superlinear (exponential) dynamics may, however, become visible over longer time horizons. While these studies refer to the development of closed software systems, it is relevant in the context of digital platforms to consider OSS. The development process of OSS has changed significantly from the traditional development of the software—above all by its generative nature and release frequencies (Keßler 2013). Godfrey and Tu (2000) studied the size and growth of the Linux kernel from 1994 to 1999 and concluded that the growth rate was superlinear. Koch and Schneider (2000) report in their study on GNOME that the size of the code base grows super-linearly and point to a correlation with the number of active programmers. Also O'Mahony (2006), in her study on Debian Gnu/Linux, showed that the growth over several releases is superlinear. Together, these studies suggest that digital platforms drawing on external developer communities could profit from rapid external innovation similar to OSS, making superlinear growth patterns likely.

Platform theory also gives indication for a superlinear growth pattern. Considering multi-sided platforms, Hagiu and Rothman (2016, p. 2) state that "once marketplaces reach a critical inflection point, network effects kick in and growth follows an exponential, rather than linear, trajectory." Although these authors do not offer empirical evidence or explanation of the term "exponential," network effect theory gives us a robust understanding of the cross-sided dynamics driving platform growth. On the one side, platform models allow a rapid scaling of the user base, almost as if "on steroids" (Huang et al. 2017). Huang et al. (2017) give the example of the WeCash platform and indicate monthly user growth rates of 21 to 158 percent. Whatsapp has grown from 0 to 419 million users in four years (Smith 2014). On the other side, third-party complements also grow in a rapid pace, since value can be captured from the additional users. This contributes to a vicious cycle of user base and value growth, important for cross-sided network effects. Song et al. (2018) indicates that there may be more complex interaction effects. However, based on previous studies on digital platforms, it is reasonable to believe that because of rapid ("exponential") user scaling, once network effects kick-in, also the overall code base would take a superlinear pattern. Thus, we expect:

H1: The growth of a digital platform takes a superlinear pattern.

Complexity of Digital Platforms

While the growth analysis of a digital platform focusses on the size of the code base, complexity analysis turns to a macro view of the structure of the platform. Several studies investigated the complexity of software systems in general, platforms, and OSS in particular. Complex systems comprise a large amount of actors whose interactions are dynamic (Simon 1962). The characteristics of digital platforms make them particularly prone towards high complexity. Their user base is exponentially growing while an ever growing extend of resources and services is being integrated (Evans and Basole 2016), effectively leading to more and more actors interacting in the platform ecosystem (Tiwana 2013; Tiwana and Kim 2015). Previously, we emphasized that digital platforms follow modular architectural designs to enable higher degrees of generativity while keeping complexity at bay (Hanseth and Lyytinen 2010; Yoo et al. 2010). Integrating more third-party developers who do not align with fixed and central governance and exchange resources in communities add to complexity. (Yoo et al. 2010). Despite the lack of control through fixed design rules and the resulting diverse course of the development, basic structural pattern can be observed (Singh et al. 2011; Um et al. 2015). While some studies have mapped the entire ecosystem of developers on platforms around several open source projects (Blincoe et al. 2015; Eck and Uebernickel 2016), this paper attempts to describe which structural collaboration patterns in a network emerge. By network, we refer to the relationships between developers and small internal projects (repositories). Based on Wasserman and Faust (1994), Singh et al. (2011) define this type of relationship between a set of actors and social events as an "affiliation network." Graphically, such affiliation network can be represented as a bipartite graph, in which both the

programmers as well as the software modules represent nodes (Singh et al. 2011). Furthermore, the bipartite network can be collapsed into a one-mode network in which nodes represent modules and a (weighted) link represents one or more developers contributing to two repositories (*A* and *B*).

Since affiliation networks do not capture the full range of platform complexity, it is worth arguing for their usefulness. We do so by turning to Conway's law (Conway 1968). In its most basic form, it holds that "the structure of code mirrors the structure of the organization" (Herbsleb and Grinter 1999). In other words, technical structures tend to reproduce communication structures of the organization (Baldwin and Clark 2000; Conway 1968; Sanchez and Mahoney 1996). Colfer and Baldwin (2016) give a comprehensive survey of how organizational links between projects or companies will correspond to the technical dependencies of the underlying system. Organizational links may consist of communication links between individuals such as co-location of developers. In short, the above hypothesis describes how, within a complex system, the network structure of the technical architecture ("what depends on what") on the one hand and the division of labor ("who does what") on the other correspond to each other, with some exceptions (Colfer and Baldwin 2016). This suggests that affiliation networks will to some extent represent the architectural design.

Based on the works of Singh et al. (2011) and Um et al. (2015), we develop the idea that structural patterns will emerge in such a network. Grounded in the work of modularity theory by Simon (1962), we know that modularization will increase the possible speed of evolution of a complex system. Different teams working with different toolsets and methodologies can develop these modules independently. According to Um et al. (2015), important effects occur over time. Their size will increase since developers add new functions continuously. Thus, modules will become more heterogeneous over time. In addition, modules will merge in larger units or clusters. These clusters consist of multiple relatively homogeneous modules having fewer links to other clusters. This effect was described by Um et al. (2015) for the content management system Wordpress. Singh et al. (2011) show structural separation in the Python project. We suspect that similar dynamics will also occur for a digital platform in a mixed (corporate developers and volunteers) setting. Thus, we posit:

H2: A digital platform will develop structural patterns mirroring important sub-communities.

Generativity of Digital Platforms

Stakeholders interacting with digital products leave digital traces (Eck and Uebernickel 2016). Some of these data act as important signals of trust and utility for the other actors (Dabbish et al. 2012; Jarvenpaa et al. 1999). Examples are the number of stars, open issues, and forks in a GitHub repository (Dabbish et al. 2012). Developers interpret this data as indicator that the respective project is active and can add value to the community (Dabbish et al. 2012). The developers in GitHub have three possibilities of code development at their disposal: create a "commit," "fork" the project or apply for a "pull request" (Dabbish et al. 2012). The project owners can edit contents of the code files directly by creating a commit to the code directly or to a separate branch (Dabbish et al. 2012). Those who do not have ownership rights but want to contribute to the code base, copy the repository ("fork"). They propose changes to the main code base in a pull request, which can be granted or rejected by the owner (Dabbish et al. 2012). The repositories, in turn, can be copied by the developers to serve as the basis for use in an independent project (Zhang et al. 2014). Therefore, a "fork" can be used as a sign of trust for participation of developers in a project (Zhang et al. 2014). This fact shows that "forks" represent active participation of third-party developers in a project. They indicate that an external party found the code potentially useful for their own use cases and applications. Therefore forks serve as externally proved and therefore strong indicators of generativity.

We develop the idea that the *activity* devoted to a software module *leads to generativity*. We define activity as the number of "open issues" in a software development project. Open issue fora enable users and developers to request new functionality (i.e., fix errors and add capabilities). Many open problems indicate a large functionality, which in turn speaks in favor of a generative development of the software module. The theoretical rationale for how activity leads to generativity is grounded in signaling theory (Connelly et al. 2011). Under asymmetric information, signals help to reduce the costs for searching good quality content. According to the theory, in a competitive situation, agents will over time learn and get able to discern true signals from false ones. Thus, the signal will over time come to reflect the true quality of the underlying good or service. It is, therefore, reasonable to suspect that a good or service having received activity signals from many people will also be selected for appropriation via generativity forking. Thus, we posit:

H3a: An increase in activity leads to an increase in generativity.

We further develop the idea that *generativity* leads to *popularity*. We define popularity as the number of developers that star (or "bookmark") a software module. The reason to suspect a link between generativity and popularity is that many forks represent a large active community. Forking allows developers to edit an original code base for their own interest. Private branches can also be merged via pull requests. More forks usually result in more frequent changes of the code base, increasing contributions. This tends to make the code more useful and adds to its quality over time. It acts as a sign of trust and can attract more users who simply want to use the code unchanged (i.e., passively), eventually increasing its popularity. Furthermore, generative forking of a software module will enable developers to build on something that has already been of high interest and to make a fresh start. Higher quality will in turn lead to a repository that will be used more and will provide additional functionality and in turn become more popular. The theoretical rationale is that developers will appreciate the increased functionality of a generative software module and will therefore signal their appreciation through starring. In consonance with this thesis, Um et al. (2018) have found a positive relation between an increased number of structural changes, a measure of contributions, and weekly downloads of a software module, a measure of popularity. In sum, we posit:

H3b: Increased generativity leads to an increase in popularity.

Methods

We conducted an empirical case study of the platform Otto.de. The evolution of the Otto.de platform is embedded in the context of a large retail company, Otto, which has transformed from a catalog shipper to an e-commerce-first company in recent years. Otto is part of the Otto Group, one of the top ten e-commerce companies in the world. Interestingly, Otto had long produced a physical catalog for mail or telephone ordering and while many such companies had disappeared, Otto was "the only traditional European mail order company to survive the digital change" (Jardine 2019, NZZ).

Background: The Case of the Catalog Company Otto and its Digital Platform

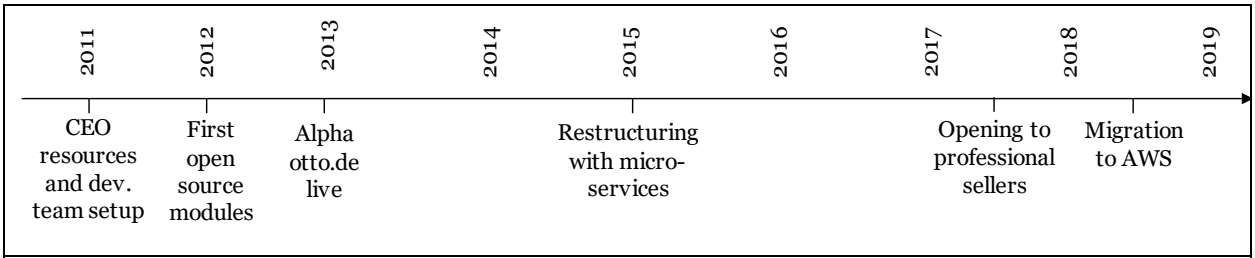

Figure 1. Timeline of Main Events of Otto's Platform Development

Figure 1 gives a brief timeline on the platform's evolution over time. In 2011, Otto's CEO/board approved the marketing unit's request to develop a new web-shop platform and *allocated resources* in the double-digit million Euro range. One of the main motivations was the insufficiency of existing systems and the need to react more quickly in a rapidly changing market place. This is why the unit aimed for a self-developed platform using agile principles. The platform project consisted of 100 staff members organized in seven interdisciplinary teams (i.e., programmers, test managers, UX and design specialists, project and product managers, etc.). These teams were organized around functional areas (search, navigation, product presentation, etc.), as detailed by Kraus and Wolter (2016). The platform development team had since then grown to its current size of 250 employees.

In 2012, Otto started to *make some of the platform's codes open source*. By involving external developers, the marketing unit aimed to increase the leverage and scale of the platform more rapidly. It also responded to the unit's architectural principles, which they had developed independently of the central IT. Among those were principles on the macro level (e.g., "RESTful architecture", or "central responsibility for data and data supply processes") and at the micro level (e.g., "buy when not core", or "common basic technologies"). By doing so, the firm attempted to ensure that redundancies remain controllable and the

architecture does not erode. Generally, open source software was thought to help to share code that several teams needed. In 2013, a first *alpha version* and a few month after that a beta version of the platform went live. It is one of the largest agile development projects in Germany in a traditional enterprise and provides insights into "scaling agile … at scale" (Kraus and Wolter 2016). In 2015, the project was *restructured using a microservices approach*. This approach allowed an "independent systems architecture" through modularization and loose coupling of deployed services (Killalea 2016) and contributed to the further scalability of the platform. In mid-2017, the platform was opened up to external sellers. Otto's new CEO at that time also positioned Otto as a platform company and built on the strength of the platform as a basis for a new corporate culture. In 2018 and 2019, they attracted 1.8 and 2 million new customers, respectively, which accounts for approximately 25% of the total number of their customers.

Data Description

Our data is the GitHub repository of the platform (https://github.com/otto-de). Past studies already confirm the usefulness of specific GitHub parameters when analyzing platform growth (e.g., Borges and Valente 2018; Mackenzie 2018). In this paper, we do not focus on the "multi-sidedness" of the platform towards professional sellers, which had just begun in 2017. Yet, the interesting aspect here is that the platform had been "opened up" to external developers, which moves it away from an internal IT platform to a model where combinatorial innovation becomes possible through the contributions of external developers writing code which is integrated into the platform on the level of particular software modules (i.e., repositories).

We extracted the entire public repository via a Python script utilizing the GitHub API; this covers panel data tracing back from the platform's external opening in August 2012 until March 2019 (6.5 years). This "digital trace data" (Eck and Uebernickel 2016) was preprocessed and resulted in a detailed overview of the platform's 65 repositories, 506 contributors, and 710,138 lines of code. The developers made 11,479 commits, i.e. minor or major changes of code in a repository in total. Additional qualitative data was collected to make sense of the quantitative data. They stem from blog posts (#55), field visits (#2), scientific articles (#4), and industry intelligence articles (#80). Several conversations with leading managers and developers of the platform unit confirmed the representativeness of the GitHub data for the project's development in general. In the following section we describe several parameters we used to operationalize our theoretical concepts that are provided by GitHub at any given developer activity.

Operationalization of Study Variables

Table 1 shows our operationalization of the study variables and gives descriptive statistics. For analyzing platform growth, we used the *absolute difference (additions minus deletions) in the lines of code* (LOC) over all considered repositories. The measure taps into the degree to which new code is added to the platform and thus captures well our understanding of growth. Our measure was appropriate because data was available at fine-grained time points for every repository, allowing tracking LOC growth in meticulous detail. As an alternative conceptualization, we used the *absolute growth in the lines of code (additions plus deletions)*, delivering consistent results. Using ordinary least squares (OLS) regression analysis, we regressed growth against time (in month).

For analyzing complexity, we used the *weighted degree* of a repository by considering interdependencies with other repositories through joint developers. This conceptualization taps into the collaborative work structure in the platform development project and captures the social component of complexity. To generate the underlying network, we firstly extracted the bipartite relations between repositories and developers and secondly collapsed this into a one-mode network of inter-repository relations via joint developers. Gephi, an open source network analysis tool, was used for visual analysis and metrics calculation. The weighted degree is the sum of all interdependencies for a repository.

Table 1. Variable Operationalization and Distribution			
Variable	**Definition**	**Operationalization**	**Distribution**
Model variables			
Growth	Measure to determine the growth of the platform	Number of additions minus deletions in terms of lines of code (count)	*Lines of code (added)*: 710,138 lines *Avg. code added per month:* 10,925.2 lines
Complexity	Scope of the platform development in terms of collaborative work on the repositories	Weighted degree of a repository in one-mode network of inter-repository relations via joint developers	*Avg. weighted degree*: 16.79 *Nodes*: 66 (61 filtered) *Edges*: 512
Generativity	Further development of the repositories by external developers	Number of forks of a repository (count)	*Avg*: 4.75; *SD*: 8.38; *Min*: 0; *Max*: 48
Popularity	The interest and satisfaction of developers with a repository	Number of stars of a repository (count)	*Avg*: 13; *SD*: 8.18; *Min*: 1; *Max*: 36
Activity	Work on unresolved problems of a repository	Number of open issues of a repository (count)	*Avg*: 1.32; *SD*: 3.37; *Min*: 0; *Max*: 15
Control variables			
Contributors	Developers working on a repository	Number of contributors of a repository (count)	*Avg*: 7.7; *SD*: 12.20 *Min*: 0; *Max*: 90
Age	Age of the repository	Number of days since the repository was initially created (count)	*Avg*: 1183.63; *SD*: 506.91 *Min*: 94; *Max*: 2,389
Weighted degree	Developers working on the considered repository and also on another repository	Avg. weighted degree of the nodes in the inter-repository network (count)	*Avg: 16.79; SD: 32.44; Min: 0; Max: 145*

To analyze generativity, we conceptualized generativity in terms of the *number of forks* of a repository. The number of forks captures the number of times developers import and further develop a repository via their own project or purpose. It thus illustrates the ability to generate new "output" (Zittrain 2008) from a particular input (repository) beyond the intention of the original creators.

We conceptualize *popularity* as the *number of stars* of a repository. Starring a repository on GitHub is a bookmarking mechanism. It signals interest or satisfaction with a repository (Borges and Valente 2018). According to a study by Borges and Valente (2018, p. 1), "three out of four developers consider the number of stars before using or contributing" to a GitHub repository. We consider the *activity* on a repository by using the *number of open issues*. This measure taps into the work on unresolved problems of a repository. Open issues represent the range of functionality (i.e., bugs and new capabilities) requested by GitHub users. They are important artifacts in a GitHub project indicating the community's activity (Dabbish et al. 2012).

We further consider three control variables. First, we use the *age* of a repository in terms of the number of days since it was initially created. Showing that generativity and popularity are independent of age would

give reason to believe that it is not the mere time passed that determines the further usages of a module. We secondly use the *number of contributors* to control for the number of active developers of a repository. Finally, we consider the number of developers jointly working on a given repository and another measured by the *weighted degree* of the nodes in the aforementioned affiliation network.

Results

Growth Analysis: A Superlinear Growth Pattern

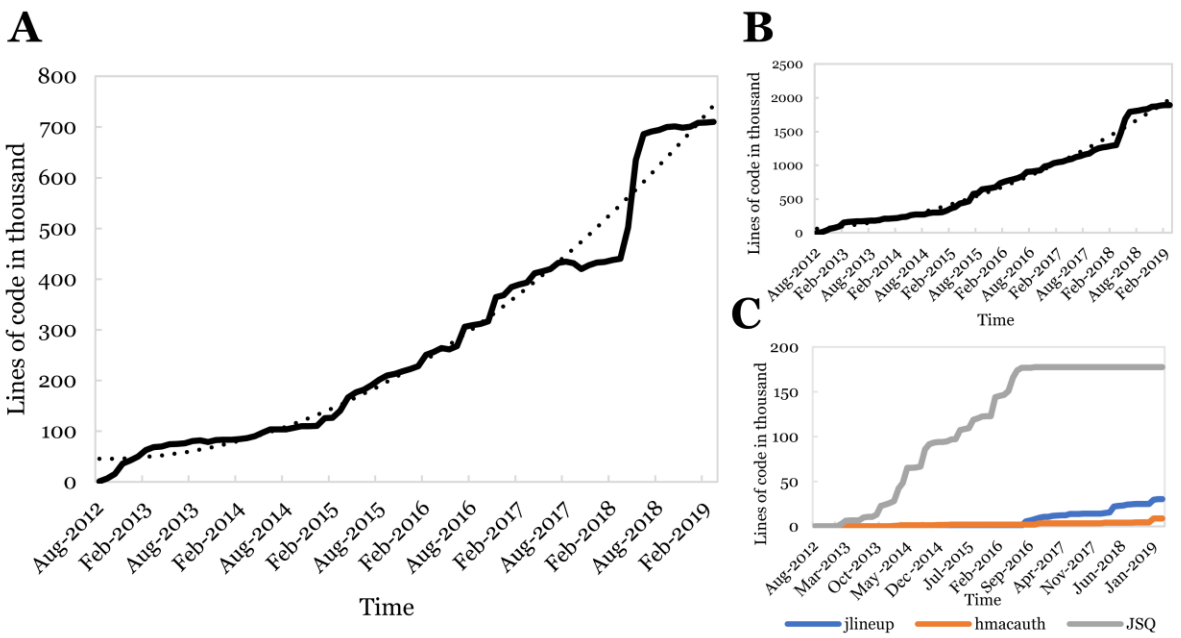

Figure 2. (A) Cumulated Growth of All Repositories Measured Using the Difference of Additions and Deletions of Lines of Code in the Github Project, (B) Alternative Growth Measurement Using Additions Plus Deletions, (C) Growth of Selected Repositories

Figure 2A shows the cumulative growth of all repositories of the Otto.de Github project measured by the difference between added and deleted code over a period of 6.5 years. Results of the growth analysis point to a stepwise growth; the sharp increases in 2015 and 2018 indicate a superlinear pattern. The dotted line indicates a polynomial regression model with degree of two indicating quadratic growth of the data with a very high coefficient of determination (R^2=0.98). The superlinearity of growth corresponds to the examples of open source software development presented in the work of Scacchi (2006). A conclusion can be drawn from the shape of the growth curve and the trend figure: the activity of the actors is linked to the creation of new repositories. No repositories were created in 2014 and commits stagnated. In 2015, however, 28 repositories were newly created and the activity increased. The pause in 2014 can be explained by the planning and development of concepts for future development.

Figure 2B and figure 2C show alternative measurements of growth and provide consistent results: Figure 2B shows the cumulative growth of repositories based on added plus deleted lines of code over the same period as figure 2A. This is important, because refactoring via deletions may also be seen as important change, giving further confirmation for our finding. Again, a polynomial regression model with quadratic growth (R^2=0.99) confirms the superlinearity of the growth pattern. Figure 2C shows the individual growth of the three largest repositories measured cumulatively via added minus deleted lines of code. For better readability, we selected a logarithmic y-axis. The repositories provide services for a tool to automatically detect design changes of a website ("jlineup"), a cryptographic message authentication for RESTful web

applications ("hmacauth"), and a messages user interface library for iOS ("JSQ"). All three repositories have been in operation since the beginning of 2016 and all are showing increasing growth rates. The analysis thus suggests that JSQ and other repositories may follow a self-similar development pattern to the overall project. Altogether, the indications presented in this and the last paragraph are thus a strong indicator for a confirmation of the hypothesis H1.

Additional qualitative analysis (i.e., blog post analysis) pointed to the following reasons for the observed development patterns: The first kink in the graph, in 2015, can be explained by a refactoring of the platform using a microservice approach. This introduced a lot of revisions as well as new code. The second kink in the graph, in 2017/18 is most likely a reaction to two events: first, in 2017 the platform was opened up to professional sellers. This required a lot of refactoring on the backbone side of the platform. Also, in 2018, infrastructure components of the platform were moved to the Amazon AWS. This required new services and a reengineering of existing functionality. Overall, the observed pattern is well aligned to the timeline in figure 1.

Complexity Analysis: Two Sub-Communities in the Platform

Figure 3 shows a network plot of the Otto.de repositories and its joint work on them as well as its degree distribution. Figure 3A displays all repositories of the Otto.de project as nodes and a connection between two nodes via an edge, if at least one developer jointly works on both repositories. The size of the nodes reflects its degree: The node is larger if there are many developers working on its corresponding repository as well as other repositories. The coloring of the nodes represents a well-known clustering approach based on the modularity of the network (Blondel et al. 2008). Modularity measures the strength of division of a network into modules, i.e. clusters. Finally, we chose the force-directed network layout Force-Atlas implemented in Gephi, which positions nodes based on laws of attraction and repulsion. It is a common method when the objective is to visually identify central or peripheral nodes and clusters. Figure 3B and figure 3C show the degree distribution and, respectively, the log-logged degree distribution. It can be seen that large hubs exist that both connect the two clusters and draw a lot of joint workload. Although the edges are clearly not independent from each other, the log-logged degree distribution gives no indication of a scale-free network (Barabasi and Albert 1999). Rather, the network plot and degree distribution indicate a "caveman network" (Cai et al. 2017). Such are created by community structures that are evolving over time and are more homogeneous in node degrees than scale-free networks.

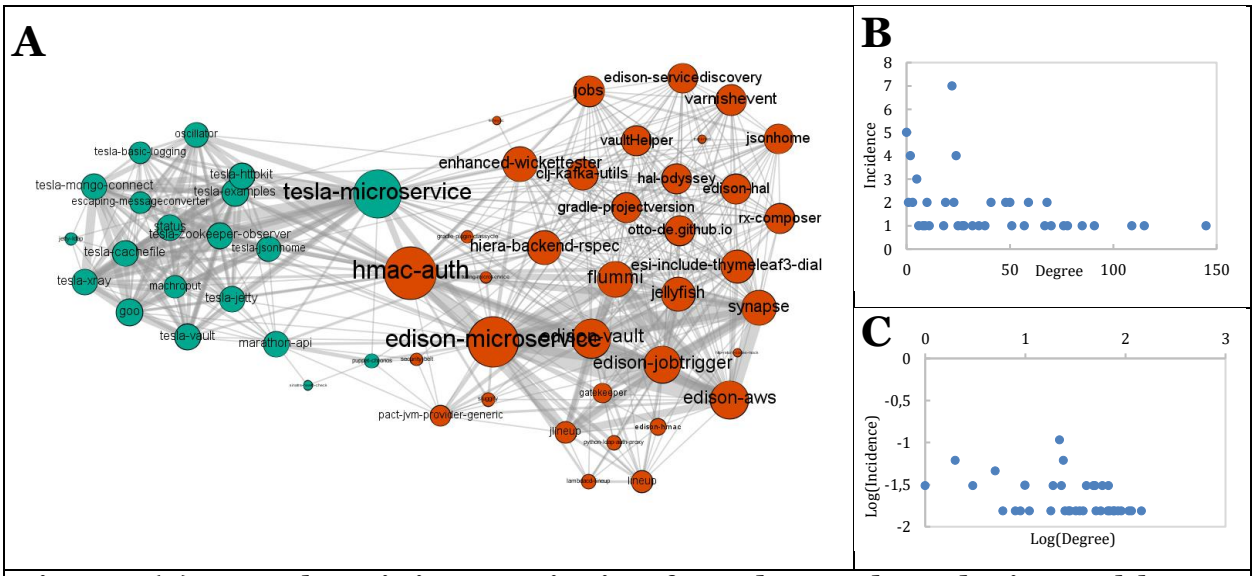

Figure 3. (A) Network Depicting Repositories of Otto.de as Nodes and Joint Work by One or More Developers as Edges, (B) Degree Distribution, (C) Log-Logged Degree Distribution

Complexity analysis pointed to the emergence of two different community clusters, which were demarcated by two main hubs (i.e., Tesla and Edison microservice). This is in consonance with the findings of Singh et

al. (2011) and Um et al. (2015) that the development of a network has a basic structural pattern. The distributed architecture of the digital platform allows the individual "building blocks" to be developed separately. Regarding sub-community analysis, we found that platform microservices in separate clusters (e.g., Tesla and Edison) are written in different programming languages. The two most widely used languages in the examined repositories are Java and Clojure. While the latter is used by most repositories in the red cluster and almost none in the green cluster, the inverse is true for Java. By adhering to standards of RESTful APIs it is possible to ease the process of transferring data. The relationship mapping also allows conclusions about the generativity and the knowledge exchange in such externally-infused projects, suggesting that more than 100 third-party developers participated. We, therefore, find indication that a digital platform will develop distinct structural patterns mirroring important sub-communities and hence that hypothesis H2 can be confirmed.

Generativity Analysis: The Influence of Activity and the Resulting Popularity

We turn to analyses regarding generativity using Spearman's Correlation Coefficient. Table 2 presents results. We find a moderate to strong associations between popularity (stars) and generativity (forks) ($r(57)=.66$, $p<.01$) as well as a weak association between activity (open issues) and generativity ($r(57)=.34$, $p<.01$). Furthermore, the correlations with contributors, age, and weighed degree are examined. Contributors correlate robustly with generativity ($r(57)=.59$, $p<.01$) and weakly with popularity ($r(57)=.40$, $p<.01$), consistent with the findings of Borges and Valente (2018). Age is very weakly associated with generativity ($r(57)=.25$, $p<.1$) and moderately with popularity ($r(57)=.41$, $p<.01$). Weighted degree is correlated weakly with generativity ($r(57)=.42$, $p<.01$) and is not correlated with popularity ($r(57)=.22$, $p>0.1$).

Table 2. Spearman Correlation Analysis

	1	2	3	4	5	6
1 Generativity	1					
2 Popularity	0.66***	1				
3 Activity	0.34***	0.35***	1			
4 Contributors	0.59***	0.40***	0.39***	1		
5 Age	0.25*	0.41***	-0.18	0.14	1	
6 Weighted degree	0.42***	0.22	0.15	0.62***	0.23	1
Mean	4.754	11.938	1.323	7.708	1181.631	32.277
Std. Deviation	8.443	25.934	3.396	12.295	510.857	31.73
Minimum	0	0	0	0	92	0
Maximum	48	182	15	90	2387	145

Note: *p<0.1; **p<0.05; ***p<0.01

To test H3a, we developed a first set of multivariate models (M1 and M2) for explaining generativity via activity. The model included age, contributors, and weighted degree as controls. Using OLS regression, model 1 and 2 are performed on generativity as outcome, first explained only by the control variables and then after addition of activity as an independent variable. Table 3 provides the regression coefficients as well as overall model diagnostics. The control variables age and contributors do not have significant impact on generativity, while the weighted degree show significant, but moderate impact. The addition of the independent variable activity does not only add a significant and valuable (0.68) effect, but also increases the explained variance of the model (from 0.43 to 0.49 respectively). Overall, the findings provide some evidence that H3a can be confirmed.

Testing H3b, Models 3, 4 and 5 use popularity (measured by stars) as the outcome variable. Model 3 reports only the control variables. Age and contributors turn out to be non-significant, while weighted degree show a moderate and significant impact. By considering generativity as an explanatory variable in Model 4 none of the control variables are significant, whereas generativity appears to be impactful (2.19) and highly significant. Also Model 4 shows a much higher explained variance compared to Model 3 (0.26 to 0.55 respectively).

Finally Model 5 considers generativity as a mediating variable of the relationship between activity and popularity. In order to establish mediation activity must influence generativity, and generativity must influence popularity. The results in Table 2 support both requirements. Furthermore, the direct effect of activity on popularity reduced in significance and value, which indicates the mediation role of generativity. In order to examine the significance of the indirect effect a bootstrapping method with 10,000 iterations was applied and found the relationship to be significant ($p<0.05$), thus further supporting generativity as a mediator (Zhao et al. 2010). Since the direct effect is not significant, while the indirect effect is, the model supports an indirect-only full mediation with unlikely omitted mediators and a consistent theoretical framework. The overall model fit is one of the highest of all examined models with $R^2=0.54$. The very slight decrease of the coefficient of determination from M4 to M5 is due to the omitted control variables in the mediation model. These findings provide evidence that H3b can be confirmed.

Table 3. OLS Regression Results

	M1	M2	M3	M4	M5
	Dependent var.: Generativity (Forks)		Dependent var.: Popularity (Stars)		
Intercepts					
	-1.83 (0.215)	-2.59 (2.06)	-9,58 (7.50)	-5.57 (5.92)	
Controls					
Contributors	0.13 (0.07)	0.11 (0.07)	0.23 (0.25)	-0.07 (0.20)	
Age	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	
Weighted degree	0.14*** (0.03)	0.12*** (0.03)	0.35*** (0.10)	0.02 (0.09)	
Independent variable					
Activity (Open issues)		0.68** (0.25)			-0.56 (0.74)
Mediation effect					
Generativity (Forks)				2.19*** (0.35)	2.36*** (0.30)
N	65	65	65	65	65
R^2	0.43	0.49	0.26	0.55	0.54
Adjusted R^2	0.40	0.45	0.23	0.52	-
F-Statistic	15.16***	14.55***	7.27***	18.58***	36.76***

Standard errors in parentheses
* significant with $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Discussion and Conclusion

The study considered growth, complexity, and generativity of a digital platform–Otto.de. The platform provides interesting incentives to study the case particularly because of a) its pivotal role in Otto's digital transformation, b) its growth pattern, and c) the underlying development principles adopted. Furthermore, it is an indicative example of a large company transforming its IT approach from an internal model to an open platform model. Considering growth, we find a quadratic pattern indicating a superlinear growth over the period mid-2012 to early 2019. This was driven by the modularization of the platform using a microservice approach as well as opening the platform to professional sellers. It thus reflects Otto's digital transformation journey and mirrors important junction points. Considering complexity, our findings indicate a structural separation of the platform into two main clusters, pointing to the possibilities that structural patterns can be reinforced over time. Given the size of the platform project, it occurs as natural that preferences and tastes are reflected in the platform. However, the coherence within the clusters appears interesting, which could indicate future splitting tendencies. Considering generativity, our analysis points

to the influence of activity on the generativity of a digital platform as well as its implications for popularity. This reflects the observation that some modules are perceived as particularly "hot" and receive corresponding attention, while others develop below average. Taken together, these findings provide a deep insight into the structure and development of a digital platform and opens up future research avenues.

Our study makes several contributions to the literature on platform evolution and generativity. First, Lyytinen et al. (2017) imply that a platform's generativity allows unbounded new functions to emerge. Specifying this finding, we find a superlinear (quadratic) growth pattern of a digital platform–Otto.de. This finding is consistent with results from Scacchi (2009), Godfrey and Tu (2000), and Koch and Schneider (2000), who studied open source software. Our study shows this in a setting in which internal developers work jointly with external open source developers. It also shows that growth is enabled by generative forking, which is dependent on an active community, and which in turn leads to a popular platform. Second, de Reuver et al. (2017) imply that sequence mining techniques will help to understand platform evolution. Our study exemplifies this in the context of a digital platform and adds insights from additional developer blog and other qualitative data analyses. Third, we also contribute to the debate on platform complexity (Tilson et al. 2013). We show the modularity structure of a digital platform and propose a technique based on mapping the collaborative work structure of developers working jointly on software modules as a network. According to the terminology of Faraj et al. (2016), this represents "online communities" collaborating on a project. Given that the structure of a code tends to mirror the structure of an organization (Herbsleb and Grinter 1999), we believe that this is a useful methodological advancement for situations in which no granular interconnection data between modules is easily accessible.

Our study was limited by the chosen approach and instrumentation. First, we considered only a single case of a digital platform and its quantitative evolution over 6.5 years. Despite the single case limitation, the longitudinal nature of our study provides us the opportunity to study the evolution of the platform, which we argue is necessary to unpack the growth, generativity, and complexity observations in our study. While we acknowledge that the single case purview of our study limits its generalizability, our findings are in consonance with observations in prior literature (Scacchi 2009, Godfrey and Tu 2000, and Koch and Schneider 2000), which suggests that our conclusions are not idiosyncratic to our case but may find relevance in other related contexts. Furthermore, our study situate its analytical focus on the platform rather than the case organization which implies that our theoretical expositions are at the platform level and can be abstracted to platforms enveloped in similar operational networks . Second, our analysis was limited to the public parts of the Otto.de repositories, giving opportunities to see whether non-public parts differ in their growth and complexity. However, talks with the head of platform development as well as other responsible managers indicate that the development of the investigated modules is representative of the overall growth dynamics. Third, our chosen methodology to study generativity only reveals associations and not the causality inherent in the arguments. Our results on the associations of generativity, popularity, and contributors as well as (non-significantly) age and popularity are consistent with Borges and Valente (2018). Future work should test for (Granger) causality using time series data of the collected activity traces.

From a practical standpoint, our study has several implications. First, it seems possible to develop and scale a company-internal digital platform in a distributed way, drawing on an external developer community. Several advantages arise such as access to a talented pool of developers and insight into the latest technologies. However, as noted elsewhere, challenges exist on how to provide value to and control such external developers (Ghazawneh and Henfridsson 2013; Tilson et al. 2010). Our study exemplifies the possibility of evolving a digital platform in a mixed corporate-volunteer model over several years. Our study suggests that managers can draw from the insights of Otto's approach to stimulate external participant's contribution. Specifically, they can do this by paying attention to the influence of activity levels on the popularityand "hotness" of a platform and leveraging this to appeal to external participant contributors. Practitioners can draw inspiration from Otto.de's example in formulating their own approach towards achieving an open platform model. The possibility for superlinear growth as shown in our study provides managers with additional incentive to do so. In terms of fostering generativity, managers may be well served by opening up their platforms for input from external contributors as well as being receptive to the new and emergent opportunities that may unfold. Specifically, a practical approach that can be gleaned from the Otto case is the idea of jettisoning an inward looking approach for combinatorial innovation, adopting an open model and embracing the emergence of generative forks. Furthermore, practitioners building a digital platform in-house may find the case description useful.

Beyond replicating our findings in other settings, future research could consider three broad directions. First, it would be interesting to consider alternative conceptualizations of generativity. Such conceptualizations could account for the value and quality of the generated new outputs/modules, e.g., how often they are used. It could also consider the scaling dynamics of the forked modules, i.e., whether they grow (sub-)linearly or superlinearly. Moreover, it would be interesting to link generativity to user base-related popularity measures such as platform usage or user satisfaction. Second, it would be fruitful to distinguish between internal and external developers. This would particularly help understand the mechanisms of disembedding platforms from local contexts. It would also contribute to the study of sustained contributions in mixed corporate-volunteer contexts. Such understanding would be relevant to understand how the e-commerce company and others collaborate online and offline with different developer and stakeholder groups working on different components of a digital platform. Finally, a richer conceptualization of platform architecture could be developed taking into account actual dependencies between modules and their interaction. This presents important research challenges as well as opportunities for novel scholarly insights.

In conclusion, we see our study as a starting point for a richer and more contextual understanding of platform evolution. We link empirical analysis of platform growth in terms of the actual code base with additional investigations of generativity and complexity. We show unprompted change at two time points (2015 and 2018), corresponding to the introduction of microservices and opening up the platform to professional sellers and demonstrating the generative potential of the digital platform. We further show how more active parts of the code base tend to be more generative and will become more popular. We also show how complex interconnections manifest in the code base through an inter-developer affiliation network and how this network begins to show structural patterns mirroring important sub-communities. This collaboration structure represents an important constraint for platform evolution. Taken together, our study draws a more nuanced picture of how generativity and complexity influence platform evolution.

References

Agarwal, R., and Tiwana, A. 2015. "Editorial—Evolvable Systems: Through the Looking Glass of IS," *Information Systems Research* (26:3), pp. 473–479.

Baldwin, C. Y., and Clark, K. B. 2000. *Design Rules, Volume 1: The Power of Modularity*, Cambridge, MA: MIT Press.

Baldwin, C. Y., and Woodward, C. J. 2009. "The Architecture of Platforms: A Unified View," in *Platforms, Markets and Innovation*, Finance Working Paper, A. Gawer (ed.), Cheltenham, UK: Edward Elgar, pp. 19–39.

Barabasi, A., and Albert, R. 1999. "Emergence of Scaling in Random Networks," *Science* (286:5439), pp. 509–512.

Blincoe, K., Harrison, F., and Damian, D. 2015. "Ecosystems in GitHub and a Method for Ecosystem Identification Using Reference Coupling," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. 2008. "Fast Unfolding of Communities in Large Networks," *Journal of Statistical Mechanics: Theory and Experiment* (2008:10), p. P10008.

Borges, H., and Valente, T. M. 2018. "What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform," *Journal of Systems and Software* (146:12), pp. 112–129.

Cai, M., Cui, Y., and Stanley, H. E. 2017. "Analysis and Evaluation of the Entropy Indices of a Static Network Structure," *Scientific Reports* (7:9340), pp. 1–10.

Colfer, L. J., and Baldwin, C. Y. 2016. "The Mirroring Hypothesis: Theory, Evidence, and Exceptions," *Industrial and Corporate Change* (25:5), pp. 709-738.

Connelly, B. L., Certo, S. T., Ireland, R. D., and Reutzel, C. R. 2011. "Signaling Theory: A Review and Assessment," *Journal of Management* (37:1), pp. 39–67.

Constantinides, P., Henfridsson, O., and Parker, G. 2018. "Platforms and Infrastructures in the Digital Age," *Information Systems Research* (29:2), iii–vi.

Conway, M. E. 1968. "How Do Committees Invent?," *Datamation* (14:5), pp. 28–31.

Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. 2012. "Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository," in *Proceedings of the ACM 2012 Conference on CSCW*, pp. 1277–1286.

Eck, A., and Uebernickel, F. 2016. "Reconstructing Open Source Software Ecosystems: Finding Structure

in Digital Traces," in *ICIS 2016 Proceedings*.
Evans, D. S., and Schmalensee, R. 2016. *Matchmakers: The New Economics of Multisided Platforms*, Brighton, MA: Harvard Business Review Press.
Evans, P. C., and Basole, R. C. 2016. "Revealing the API Ecosystem and Enterprise Strategy via Visual Analytics," *Communications of the ACM* (59:2), pp. 26–28.
Faraj, S., Jarvenpaa, S. L., and Majchrzak, A. 2011. "Knowledge Collaboration in Online Communities," *Organization Science* (22:5), pp. 1224–1239.
Faraj, S., von Krogh, G., Monteiro, E., and Lakhani, K. R. 2016. "Special Section Introduction—Online Community as Space for Knowledge Flows," *Information Systems Research* (27:4), pp. 668–684.
Fürstenau, D., Anisimova, D., Masak, D., Rothe, H., Schulte-Althoff, M. (2019). The Digital Platform Otto.de: A Case Study of Growth, Complexity, and Generativity. In T. Ludewig, V. Pipek (Eds.) *Proceedings of the 14th International Conference on Wirtschaftsinformatik (WI 2019)* (pp. 919-922), Siegen, Germany.
Gawer, A. 2014. "Bridging Differing Perspectives on Technological Platforms: Toward an Integrative Framework," *Research Policy* (43:7), pp. 1239–1249.
Ghazawneh, A., and Henfridsson, O. 2013. "Balancing Platform Control and External Contribution in Third-Party Development: The Boundary Resources Model," *Information Systems Journal* (23:2), pp. 173–192.
Godfrey, M. W., and Tu, Q. 2000. "Evolution in Open Source Software: A Case Study," in *Proceedings 2000 International Conference on Software Maintenance*, pp. 131–142.
Hagiu, A., and Rothman, S. 2016. "Network Effects Aren't Enough," *Harvard Business Review* (94:4).
Henfridsson, O., and Bygstad, B. 2013. "The Generative Mechanisms of Digital Infrastructure Evolution," *MIS Quarterly* (37:3), pp. 907–931.
Herbsleb, J. D., and Grinter, R. E. 1999. "Splitting the Organization and Integrating the Code: Conway's Law Revisited," in *Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No.99CB37002)*.
von Hippel, E. A., and von Krogh, G. 2003. "Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science," *Organization Science* (14:2), pp. 209–223.
Huang, J., Henfridsson, O., Liu, M. J., and Newell, S. 2017. "Growing on Steroids: Rapidly Scaling the User Base of Digital Ventures Through Digital Innovation," *MIS Quarterly* (41:1), pp. 301–314.
Jarvenpaa, S. L., Tractinsky, N., and Saarinen, L. 1999. "Consumer Trust in an Internet Store: A Cross-Cultural Validation," *Journal of Computer-Mediated Communication* (5:2).
Keßler, S. 2013. *Anpassung von Open-Source-Software in Anwenderunternehmen*, Wiesbaden: Springer.
Killalea, T. 2016. "The Hidden Dividends of Microservices," *Communications of the ACM* (59:8), pp. 42–45.
Koch, S., and Schneider, G. 2000. *Results from Software Engineering Research into Open Source Development Projects Using Public Data. [Vienna University of Economics and Business Administration]*, (22).
Kraus, S., and Wolter, P. 2016. "Scaling Agile @ Otto - Learning at Scale: Agilität Als Organisatorische Herausforderung," *OBJEKTspektrum* (4), pp. 20–24.
Lehman, M. 1980. "Programs, Life Cycles, and Laws of Software Evolution," *IEEE Proceedings* (68:9).
Lehman, M., Ramil, J., Wernick, P., and Perry, D. 1997. "Metrics and Laws of Software Evolution Process Improvement," *Growth Lakeland*, pp. 20–32.
Lyytinen, K., Sørensen, C., and Tilson, D. 2017. "Generativity in Digital Infrastructures," in *The Routledge Companion to Management Information Systems*, R. D. Galliers and M.-K. Stein (eds.), London: Routledge.
Mackenzie, A. 2018. "48 Million Configurations and Counting: Platform Numbers and Their Capitalization," *Journal of Cultural Economy* (11:1), pp. 36-53.
O'Mahony, S. 2006. "Developing Community Software in a Commodity World," in *Frontiers of Capital*, Duke University Press, pp. 237–266.
Parker, G., Van Alstyne, M., and Jiang, X. 2017. "Platform Ecosystems: How Developers Invert the Firm," *MIS Quarterly* (41:1), pp. 255–266.
Parker, G., van Alstyne, M. W., and Choudary, S. P. 2016. *Platform Revolution: How Networked Markets Are Transforming the Economy--and How to Make Them Work for You*, New York, NY: Norton & Company.
de Reuver, M., Sørensen, C., and Basole, R. 2017. "The Digital Platform: A Research Agenda," *Journal of Information Technology* (33:2), pp. 124-135.

Rochet, J.-C., and Tirole, J. 2003. "Platform Competition in Two-Sided Markets," *Journal of the European Economic Association* (1:4), pp. 990–1029.
Sanchez, R., and Mahoney, J. T. 1996. "Modularity, Flexibility, and Knowledge Management in Product and Organization Design," *Strategic Management Journal* (17:S2), pp. 63–76.
Scacchi, W. 2004. *Software Evolution and Feedback: Theory and Practice*, N. H. Madhavji, J. Fernandez-Ramil, and D. E. Perry (eds.), Chichester: John Wiley & Sons Ltd., pp. 181–205.
Scacchi, W. 2009. *Understanding Requirements for Open Source Software*, Springer, Berlin, Heidelberg, pp. 467–494.
Scacchi, W. (n.d.). "Understanding Open Source Software Evolution," in *Software Evolution and Feedback: Theory and Practice*, N. H. Madhavji, J. C. Fernández-Ramil, and D. E. Perry (eds.), New York: Wiley.
Simon, H. A. 1962. "The Architecture of Complexity," *Proceedings of the American Philosophical Society* (106:6), pp. 467–482.
Singh, P. V., Tan, Y., and Mookerjee, V. 2011. "Network Effects: The Influence of Structural Capital on Open Source Project Success," *Mis Quarterly* (35:4), pp. 813–829.
Smith, D. 2014. "The Facebook Effect: WhatsApp Is Well On Its Way To A Billion Users," *ReadWrite*.
Song, P., Xue, L., Rai, A., and Zhang, C. 2018. "The Ecosystem of Software Platform: A Study of Asymmetric Cross-Side Network Effects and Platform Governance," *MIS Quarterly* (42:1).
Staykova, K. S., and Damsgaard, J. 2017. "Towards an Integrated View of Multi-Sided Platforms Evolution," in *ICIS 2017 Proceedings*.
Tilson, D., Lyytinen, K., and Sørensen, C. 2010. "Digital Infrastructures: The Missing IS Research Agenda," *Information Systems Research* (21:4), pp. 748–759.
Tilson, D., Sørensen, C., and Lyytinen, K. 2013. "Platform Complexity: Lessons from the Music Industry," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, pp. 4625–4634.
Tiwana, A. 2013. *Platform Ecosystems: Aligning Architecture, Governance, and Strategy*, Burlington, MA: Morgan Kaufmann.
Tiwana, A., Konsynski, B., and Bush, A. A. 2010. "Research Commentary - Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics," *Information Systems Research* (21:4), pp. 675–687.
Um, S., Kang, M., Hahn, J., and Yoo, Y. 2018. "Popularity and Competition in a Digital Platform Ecosystem: A Network Perspective," in *ICIS 2018 Proceedings*.
Um, S., Yoo, Y., and Wattal, S. 2015. "The Evolution of Digital Ecosystems: A Case of WordPress from 2004 to 2014," in *ICIS 2015 Proceedings*.
Wareham, J. D., Fox, P. B., and Cano Giner, J. L. 2014. "Technology Ecosystem Governance," *Organization Science* (25:4), pp. 1195–1215.
Wasserman, S., and Faust, K. 1994. *Social Network Analysis: Methods and Applications*, (19th ed., Vol. 8), Structural Analysis in the Social Sciences, Cambridge: Cambridge Univ. Press.
Wu, Y., Kropczynski, J., Shih, P. C., and Carroll, J. M. 2014. "Exploring the Ecosystem of Software Developers on GitHub and Other Platforms," in *CSCW 2014 Proceedings*, Baltimore, MD: ACM, pp. 265–268.
Yoo, Y. 2013. "The Tables Have Turned: How Can the Information Systems Field Contribute to Technology and Innovation Management Research?," *Journal of the Association for Information Systems* (14:5), pp. 227–236.
Yoo, Y., Boland, R. J., Lyytinen, K., and Majchrzak, A. 2012. "Organizing for Innovation in the Digitized World," *Organization Science* (23:5), pp. 1398–1408.
Yoo, Y., Henfridsson, O., and Lyytinen, K. 2010. "Research Commentary - The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research," *Information Systems Research* (21:4), pp. 724–735.
Zhang, Z., Yoo, Y., Zhang, B., Wattal, S., and Kulathinal, R. 2014. "Generative Diffusion of Innovations and Knowledge Networks in Open Source Projects," *ICIS 2014 Proceedings*.
Zhao, X., Lynch, J. G., and Chen, Q. 2010. "Reconsidering Baron and Kenny: Myths and Truths about Mediation Analysis," *Journal of Consumer Research* (37:2), pp. 197–206.
Zittrain, J. 2006. "The Generative Internet The Harvard Community Has Made This," *Harvard Law Review* (2006:1974), pp. 1974–2040.
Zittrain, J. 2008. *The Future of the Internet - And How to Stop It*, London: Yale University Press.