weizenbaum
institut

WEIZENBAUM SERIES \22

WORKING PAPER \

FEBRUARY 2022 \

Alexander Ziegler

The Tech Company

On the neglected second nature of platforms

The Tech Company

Alexander Ziegler \ ISF Munich \ alexander.ziegler@isf-muenchen.de

ISSN 2748-5587 \ DOI 10.34669/WI.WS/22

EDITORS: The Managing Board members of the Weizenbaum-Institut e.V.
Prof. Dr. Christoph Neuberger
Prof. Dr. Sascha Friesike
Prof. Dr. Martin Krzywdzinski
Dr. Karin-Irene Eiermann

Hardenbergstraße 32 \ 10623 Berlin \ Tel.: +49 30 700141-001
info@weizenbaum-institut.de \ www.weizenbaum-institut.de

TYPESETTING: Roland Toth, M.A.

WEIZENBAUM INSTITUTE: The Weizenbaum Institute for the Networked Society – The German Internet Institute is a joint project funded by the Federal Ministry of Education and Research (BMBF). It conducts interdisciplinary and basic research on the changes in society caused by digitalisation and develops options for shaping politics, business and civil society.

This work has been funded by the Federal Ministry of Education and Research of Germany (BMBF) (grant no.: 16DII121, 16DII122, 16DII123, 16DII124, 16DII125, 16DII126, 16DII127, 16DII128 – "Deutsches Internet-Institut").

Abstract

The unprecedented rise of startups such as Google or Amazon has spurred an ongoing debate on the conceptualization of the corporate model these firms represent. Thus far, attention has centered on the analysis of their product and market strategies highlighting their platform nature as common feature and its defining characteristic. By applying and scaling the platform business model, these companies have been able to capture value created outside the firm. The focus on the platform nature and the evolution of their external ecosystems, however, has left the work that is done inside these companies to create and provide online platforms largely unnoticed. Against this background, the article seeks to contribute to the debate by analyzing the inner mode of production as an essential component of their corporate model. The second nature of online platform firms, it is argued, is that they are tech companies. Building on this, the article aims to reconstruct how as tech companies they have learned and perfected to continuously develop and operate the Internet applications that power their online platforms at global scale.

Table of Contents

1 Introduction

Large online platforms such as Google, Amazon, Alibaba and Tencent but also younger companies such as e-commerce technology provider Shopify, communications service Twilio, payment service Stripe or ByteDance with its social network TikTok are considered key actors of the digital transformation of economy and society. Many of these, who were founded as small startups on the West Coast of the United States, have quickly become important commercial actors and, in the process transformed the Internet from a primarily academic network into a global "information space" (Boes and Kämpf, 2007; Will-Zocholl, 2021). Funded by venture capitalists, these firms have disrupted more and more industries, starting with retail, advertising, media, film, music, and IT (Kenney et al., 2021).[1] With the proliferation of the Internet of Things, these companies as well as many new startups are entering yet more industrial branches, attempting to reorganize value chains and industry structures and challenging the incumbent firms for power in the sector (Ziegler, 2020; Butollo and Schneidemesser, 2021).

Against this background, there has been an ongoing debate on the conceptualization of the corporate model these companies represent, the reasons behind its rapid commercial success and its implications for society (Frenken and Funfschilling, 2020). This debate spans over different research disciplines from strategic management and organization studies to economic geography and sociology. Thus far, attention has centered on their product and market strategies. Highlighted as a common feature of their corporate model and seen as key to their commercial success is the fact that many of these companies are owners of widely used platforms (Kenney and Zysman, 2016; Parker et al., 2016; Srnicek, 2017; Cusumano et al., 2019). They have either built transaction platforms and transformed the market logic in many branches by accumulating network effects and monetizing their intermediary position. Or they have created innovation platforms, which provide building blocks for other businesses, and cultivated the "ability to earn profits from a captive installed base of customers" (Shapiro and Varian, 1998: 150).[2] By applying and scaling the platform business model within this spectrum, these companies have harnessed economies of scope (Gawer, 2014) and created new organizational forms which are able "to co-opt value-creating activities that are not part of the firm" (Stark and Pais, 2020: 48). Some of them have occupied de facto monopoly positions in their respective markets and realize monopoly rents (Langley and Leyshon, 2017; Staab, 2019; Peck and Phillips, 2020). To emphasize this defining characteristic of their product and market strategies, the companies have been referred to as, e.g., "matchmakers" (Evans and Schmalensee, 2016), "proprietary markets" (Staab, 2019) or "online platform firms" (Kenney et al., 2021; Stallkamp and Schotter, 2021).

Companies like Google, Facebook, or Stripe, however, were neither born as global platforms nor do their Internet-based transaction or innovation platforms develop, operate, and maintain themselves automatically. By focusing on the platform nature of these companies and the evolution of their external platform ecosystems, the work that is done inside these companies to create and provide Internet applications at global scale has largely gone unnoticed and the "employees at the platform's core" (Stark and Pais, 2020: 51) have been rendered invisible (Dorschel, 2022). From a sociology of work perspective, therefore, the conceptualization of their corporate model and an analysis of its commercial prowess which is based primarily on its product and market strategies is not exhaustive. The main argument brought forward in this article is that in order to deepen our understanding of the corporate model of these companies, it is necessary to complement the perspective on their product and market strategies and analyze their inner mode of production as an essential component. The second

nature of online platform firms, it is argued, is that they are tech companies.[3] As tech companies they have learned and perfected how to continuously develop, operate, and monetize Internet applications powering their online platforms at global scale. By taking into account the "tech company" providing the platform, it can be shown that platform firms not only capture value created *outside* the firm but also *inside* the firm – in fact, the latter serves as an indispensable presupposition to the first.

The remainder of the paper is structured as follows. In a first step, inspired by concepts from the research tradition of German industrial sociology a basic definition of the tech company as the second nature of platform firms is presented. The tech company is understood as a commercial strategy which centers around the continuous development, operation, and monetization of Internet applications. In a second step, the paper focuses on the specific capabilities which have been developed for the realization of this strategy. In a historical reconstruction of the formation of the first tech companies, it is shown that through the development of new IT-based machine systems within these startups a new technological-organizational foundation has been created upon which Internet applications could move to the center of commercial strategies. In a third step, the central results are summarized, and further research is outlined.

The basis for this conceptual paper is a comprehensive analysis of empirical material, which is continuously expanded and continued. On the one hand, it is based on evaluations of two research stays in Silicon Valley in 2015 and 2017 together with colleagues at ISF Munich. During these research stays, numerous qualitative interviews with employees and experts were conducted in startups, platform firms, and the research units of established companies located in Silicon Valley (e.g., from the automotive industry and the IT industry). This was flanked by background discussions with various representatives from the innovation system (e.g., venture capital funds).[4] On the other hand, it is based on the study of an extensive corpus of secondary literature and primary sources, such as blog posts, publicly available interviews, communication in forums (e.g., Github) and podcasts, oral history documents provided by the Computer History Museum and expert lectures on the topic.

2 Tech Companies: Analyzing the Inner Mode of Production of Platform Firms

When examining the existing research on platform firms, it is noticeable that their strategies were predominantly analyzed in a coagulated form as strategy patterns – at a time when platform firms such as Google or Amazon had already risen to the ranks of the most valuable companies in the world and had established the platform as organizational form in their respective value domains. From this starting point, it is often forgotten, however, that in the early days of the World Wide Web, patterns for the design of commercial strategies existed at best in rudimentary form.[5] None of these startups could determine in advance whether and how the Web could be embedded in commercial strategies (McCullough, 2018; O'Mara, 2019). Moreover, the prevalent ex-post perspective only partially explains why, e.g., Google prevailed with its Web search application over competing applications from Altavista, LookSmart, Inktomi, or Yahoo, or why the social network Facebook displaced other social networks such as Friendster or MySpace, even though the latter initially had significantly higher user numbers and thus first-mover advantages as well as a greater volume of capital at their disposal. Overall, the

discussion on platforms sometimes leaves the impression that any venture with sufficient financial resources and access to "digital infrastructures of cloud computing" (Grabher and König, 2020: 105) could implement the platform model without friction. Not least, a look at the diverse efforts to adapt the strategy patterns in established companies and their initiatives, for example, to build and scale platforms for industrial branches in the context of industry 4.0 or the financial sector, show that this is by no means the case (Ziegler, 2020; Butollo and de Paiva Lareiro, 2020).

These observations illustrate that a crucial dimension in the analysis of the corporate model of these companies has remained understudied. Their product and market strategies did not emerge on the drawing board but were developed in complex and messy social processes. Their conception, practical testing, inspection, adaptation, consolidation, and scaling is the result of the ongoing work of a multitude of people. As Tarnoff and Weigel (2020: 3) point out, platforms do not emerge and scale themselves on their own, "platforms are made by people". The skills required for this undertaking are materialized in the technologies, architectures, organizational concepts, documented knowledge bases, practices, and process models developed by these companies. But they are also inscribed in the "experience and practical knowledge" (Böhle, 1994) of tech workers and continuously reflected upon and developed in a complex innovation network of developer communities, universities, startup incubators, venture capital funds, publishers, and analysts (Kenney, 2000). In addition to strategy patterns such as platform concepts, it is precisely these dynamic dimensions that constitute an essential characteristic of the corporate model of a company from a sociology of work perspective, as Altmann and Bechtle (1971) have pointed out. To be able to grasp these dynamic dimensions of platform firms it is, therefore, proposed to reboot the analysis of their corporate model and center it on the social process of providing the online platform, instead of using the external view of the platform as a starting point.

In this perspective, the question arises whether there is a structural commonality regarding the inner mode of production of such heterogeneous companies that have prospered in the Internet era like Google, Amazon, Snowflake with its applications for big data analytics, Netflix[6] with its streaming service, Atlassian with its solutions for project management, Klarna with its payment services, or Celonis with its solutions for business process optimization. While operating in different value domains, these and many other companies share a common trait: *their inner mode of production centers around the continued development, operation, and monetization of Internet applications.* Big data analytics, streaming services, project management solutions, payment services and business process optimization solutions are all based on Internet applications. This distinguishes their corporate model from all other corporate models hitherto known. To grasp this structural commonality in their inner mode of production it is proposed to refer to them as tech companies.

Building on this, online-to-offline platforms such as the mobility service Uber, the accommodation broker Airbnb or the Chinese delivery service Meituan can also be considered tech companies because their inner mode of production also *centers* around the provisioning of Internet applications and cannot be implemented without them. The fact that their inner mode of production is *centered* around Internet applications does not, as these examples show, preclude these companies from extending their operations to fields outside the Internet, as illustrated not least by Amazon's extensive network of logistics centers. But it also does not mean that all the established companies that built up a web presence in the 1990s have become tech companies as a result. Rather, the differentia specifica that turns a company with an "offline footprint" into a tech company is that its "offline footprint" is shaped around the Internet application and not vice versa.

In this respect, tech companies differ from industrial companies, whose strategies center around the machine system-based production and the sale of

material goods (Ziegler, 2020: 15). They can also be distinguished from the companies in the classic software industry such as Microsoft, Oracle, SAP, or the industrial software business of Siemens. These ventures developed software, handed it over to the customer in a package, who installed it on his computer himself or whose IT department customized it to the specific company requirements, if necessary, with the help of the software company or system integrators, and operated it in his own data center or that of an IT service provider. Tech companies instead are characterized by the fact that they use the Internet as an operating system for their applications. They generate "money *with* software rather than *from* software" (O'Grady, 2015: 27; author's emphasis). As will be shown below, the moment companies make their core applications available on the Internet, both the requirements they place on their software and the way they deal with it change fundamentally.

Emphasizing the dynamic dimensions of the corporate model points to the fact that being a tech company isn't a property innate to a company. Rather, it is a characteristic that, to use Marx's words, is "essentially practical" (Marx, 1975: 5) and can be learned. This is demonstrated not least by the prominent examples of Apple, Microsoft, Intuit, or Adobe which have transformed themselves from classic computer hardware or software companies whose strategy centered around the development and sale of PCs and software into tech companies (O'Grady, 2015), while others such as the German software company SAP, e.g., do not yet seem to have succeeded in taking this step in its entirety. Following the basic definition presented, the question therefore arises as to what specific capabilities have been developed on the U.S. West Coast to be able to center a commercial strategy around the development, operation, and monetization of Internet applications. This question will be the focus of the following analysis. In doing so, it will not be possible to explore all dimensions of these capabilities. The focus will be on the historical reconstruction of the emergence of a new technological-organizational foundation for Internet applications upon which they could move to the center of commercial strategies.

3 In Search of the Foundations of Online Platforms

In many analyses of the development of startups and companies in the Internet economy, one aspect of their strategies is simply taken for granted and not considered worthy of special consideration: These companies permanently provide complex software- and data-based applications for private or business customers on the Internet, which are constantly being further developed.[7] Their applications do not only have to be capable of dynamically absorbing peak loads, but also of keeping pace with rapid business growth. The fact that this is an essential building block of their commercial strategies usually only becomes apparent when it starts to falter. There are numerous examples of this, ranging from the early days of the World Wide Web to the present day. When the Corona pandemic broke out at the beginning of March 2020, e.g., the collapse of the Internet application of the U.S. fintech Robinhood, which enables its customers to trade stocks free of charge via an app, meant that they could only watch their portfolios lose value as prices crashed (Popper and Siegel, 2020).[8] Due to the immediacy of the Internet, even the smallest downtime or data loss can directly have a significant functional and financial impact and jeopardizes customers' trust in the solutions. But how have startups – at least initially still small – succeeded in scaling Internet-based applications dynamically, keeping them highly available globally with hundreds of millions of simultaneous accesses, updating and upgrading them

permanently while operating them cost-effectively? The search for answers to this question leads back to the early days of the World Wide Web.

3.1 Warehouse-Scale-Computing: The construction of large-scale distributed systems

The ambition to provide not only largely static websites but also complex interactive applications via the Internet emerged in the New Economy in the mid-1990s. At that time, the implementation of such visions still ran up against the technological limits imposed by narrowband data transmission rates and Internet access via telephone modems. In the wake of the speculative euphoria, however, much of the enormous capital advances alongside dot.com startups flowed into telecommunications corporations, which invested that money in building broadband access (McCullough, 2018). In much the same way that in 1840s Britain the stock market boom had accelerated the development of a widespread rail network, the dot.com bubble helped lay a new Internet infrastructure in developed countries (Janeway, 2018).

Compared to Internet access with telephone modems of the Web 1.0 era, data transfer rates in broadband networks increased sharply. In conjunction with the spread of wireless networks, this paved the way for new forms of Internet usage. Both the remaining startups and new startups sought to leverage this new Internet infrastructure. Whereas the costs of providing applications via the Internet had played a negligible role for many startups at the height of the speculative euphoria, as they could be serviced from the surplus of private venture capital seeking investment, this influx of venture capital had now ebbed away for the time being. Therefore, the remaining and new startups were intensively engaged in implementing new strategies for the efficient design of their IT infrastructures.

In this constellation, pioneers such as Google or Amazon implemented a significant basic innovation: Instead of meeting increasing requirements[9] on Internet applications by purchasing powerful, expensive "high-end servers," they switched to interconnecting cheaper but also significantly more error-prone mass-produced servers to form gigantic horizontally scalable server clusters (Brewer, 2001; McMillan and Metz, 2012; O'Grady, 2014). In these distributed systems, servers constantly exchange messages with each other so that, e.g., computational processes can be accelerated by dividing them among many different server nodes and executing them in parallel before aggregating the results again. After having been in use for quite some time before mainly in the research community, the startups took the principles of "distributed computing" to the extreme in terms of software and hardware (Burns 2018). They no longer, e.g., deployed the processors with the highest peak performance, but used the processors that achieved the highest performance relative to price per unit instead (Barroso et al., 2003). By these and numerous other measures, the startups managed to improve the price-performance ratio of their IT infrastructures drastically. Observers estimated that in 2005, e.g., Google generated three times as much computing power as its competitors for every dollar spent (Vise and Malseed, 2005: 88).

No blueprint existed for building these horizontally scalable distributed IT infrastructures. Rather, a comprehensive learning process began in the startups that continues to this day. In the process, these startups faced not only historically novel engineering problems, but also the most demanding challenges for computer science in dealing with distributed systems (Helland, 2016; Killalea, 2020). Work on these tasks attracted numerous scientists, especially since it did not have to be simulated under laboratory conditions but could be done with real existing systems. With high regularity, problems had to be solved that no one had ever faced before (Gray, 2006). Key questions were how to create reliable and powerful large-scale systems from the error-prone and unreliable individual components (Vogels, 2009) or how to keep the latency times of interactive Internet applications as

low as possible (Dean and Barroso, 2013). While on the one hand the systems had to be kept running, on the other hand permanent experiments were carried out to find and exploit optimization potential. The goal was to tweak criteria such as scalability, server utilization, application availability and latency times on the one hand and to keep costs as low as possible on the other.

In this context, startups, partly in collaboration with universities, developed new technologies and concepts for the design of distributed computer clusters, e.g., consistency models and virtualization technologies (Killalea, 2008; Vogels, 2009). As a basic requirement, systems were to be designed fault-tolerant from the ground up. Individual server failures were deliberately factored into the design and to be compensated for in real time by redundant servers to have as little impact as possible on the performance and availability of the applications (Robbins et al., 2012). Jim Reese, head of IT operations at Google at the time, describes the procedure as follows:

> We built capabilities into the software, the hardware and the network – the way we hook them up, the load balancing, and so on – to build in redundancy, to make the system fault-tolerant. (cit. a. Levy, 2011: 184).

For cost reasons, but also because many solutions simply did not exist on the market, the startups predominantly used available free open source technologies for linking and operating the server clusters, e.g., operating system software such as Linux, web servers such as Apache, database systems such as MySQL and upgraded them for their purposes (Vogl, 2020).[10] Furthermore, they also developed a number of software-based technologies that were specifically designed to meet the requirements of distributed systems. These include distributed database systems (NoSQL) (Ghemawat et al., 2003; DeCandia et al., 2007), load balancers that, e.g., assign user requests to different geographically located server clusters, software tools for configuration management and system monitoring, and many other components of "cluster-level infrastructure software" (Barroso et al., 2013: 21). Much of the repetitive work, e.g., the setup, configuration and, monitoring of servers, that initially had to be done manually by engineers, could thus be automated (Beyer et al., 2016). The tools aimed at making the distributed systems function like a single computer. Also, the software tools for big data analyses that enabled startups to extract and harness the "behavioral surplus" (Zuboff, 2019: 65) from application logs were designed from scratch for distributed systems. Google, for example, developed the "MapReduce" algorithm to analyze its users' search patterns and website hits, allowing computations on petabyte-scale data sets to be parallelized and distributed across many nodes. [11]

The spectrum of measures expanded steadily. Early on, companies such as Google, Amazon and Facebook began to design the equipment used in their data centers themselves, e.g., their servers and routers, and to have them produced at low prices by contract manufacturers in Asia, following the example of the "fabless" (Lüthje, 2001: 238) manufacturers from semiconductor industry. In the meantime, they are achieving gigantic economies of scale with this production model and have completely changed the market for server hardware, previously dominated by manufacturers such as Sun Microsystems, IBM or HP (McMillan and Metz, 2012). After Google had initially placed its self-assembled "server racks" in the data centers of commercial operators such as Exodus in rented areas next to the server systems of other Internet companies, the startup quickly moved to build its own data centers in anticipation of exponentially increasing demands, but also in awareness of the strategic importance of its distributed server clusters. According to Jim Reese, they realized that "our true advantage was actually the fact that we had this massive parallelized redundant computer network (...) and we realized that maybe it's not in our best interests to let our competitors know" (cit. a. Levy, 2011: 198).

Google's employees also designed the data centers from scratch, continuously developing them further and designing them to minimize costs while increasing performance – e.g., by reducing energy consumption for cooling (Barroso et al., 2013). In addition to building these "digital cathedrals" (Mills, 2020), whose energy consumption continues to break new records despite all the measures taken to reduce it (Greenpeace, 2017), the infrastructure for transporting the data also came into focus. Google and others began buying up fiber optic networks (Levy, 2011) and laying submarine cables (Satariano, 2019). As a result of ever-increasing amounts of investment, these privately owned "global informatics infrastructures" (Haug, 2020: 24) continue to expand. They form the first building block of the technological-organizational foundation of tech companies.

3.2 Microservices and DevOps: Modularity as architectural principle of Internet applications

Complementary to the development of large-scale distributed systems, the architecture of Internet applications became the focus of the startups' innovation efforts (Baldwin and Clark, 2000). In Web 1.0, an architecture model dominated in which applications were designed as a "single unit" (Lewis and Fowler, 2014). Although they had different layers, e.g., a client-side user interface, a server-side application, and a database, the components of the application were very tightly coupled. The entire visualization, business logic, and all functionalities were mostly contained in the server-side application. For applications designed in this way, the technical term "monolith" has become established (ibid.).

As a result of business growth, however, the applications were becoming increasingly complex systems. The engineers in the startups were coming up against limits, maintaining and scaling these software monoliths with their ever more gigantic databases. On the one hand, to add new functionalities to the application, large parts of the entire application had to be rewritten again and again. This involved tedious coordination efforts between the teams and slowed down the speed of the further development of the applications. On the other hand, to scale individual processes of the application, the entire application always had to be scaled to handle the load (Vogels, 2019). Consequently, the tight interdependencies in the application's architecture not only affected the deployment of new functionalities and the scaling of the application but also hindered practices such as updating, upgrading and code refactoring, accruing "technical debt" (Winters et al., 2020).[12] As the demands on the applications grew, these problems multiplied.

To keep pace with business growth at Amazon, management and developers took the decision for a major revision of their e-commerce application starting around 2001. The goal was to embed scalability as a design principle in the application's architecture (Vogels, 2006). The monolithic architecture of the application was modularized and split into many separate loosely coupled software services (Jamshidi et al., 2018: 25).[13] Rob Brigham, who as an IT-Manager was involved in the process, described it as follows:

> We went through the code, and pulled out functional units that served a single purpose, and we wrapped those with a web service interface. For example, there was a single service that rendered the "Buy"-button on the retailer's product detail pages. Another had the function of calculating the correct tax during checkout. (cit. a. Fulton, 2015)

From then on, each of these services comprised an element of the web store's business logic together with its required data sets and was hidden behind an application programming interface (API). The individual services could communicate with the other services solely via these interfaces (Yegge, 2011). As soon as the Amazon website is visited by a user, application servers retrieve the functionalities and information of the individual services via HTTP-based network protocols and aggregate them, e.g., to form the home page.

Just as with distributed systems, in the following years, many software tools and components for modularized applications have been and are being developed in several waves at Amazon as well as in many other startups and open source projects. Among them are mechanisms for the automatic discovery of services, program libraries for fault-tolerant communication between services, and new resource-saving virtualization technologies at the application level such as containers (Jamshidi et al., 2018). The new architectural pattern has the advantage that individual services of the application can be further developed, operated, and scaled largely independently of the other services.[14] For example, if the demand for a specific function of an application increases, only the service that includes this functionality can be scaled (e.g., the shopping cart in the case of a discount promotion) without having to scale the entire architecture of the application. This can result in huge resource, time, and cost savings. In total, Amazon is reported to have invested approximately $1 billion over six years in "re-architecting" its e-commerce application (Kim, 2018: 132). In addition, the application interfaces that define how service functionality and data can be accessed can, in principle, be made available to external developers via the Internet for application development (Yegge, 2011). From an engineering design perspective, a modular technological architecture, therefore, is regarded as a "structural commonality" of most platforms (Gawer, 2014: 1242).

The modularization of the application created the basis for a comprehensive transformation of the engineering organization at Amazon. This transformation was by no means smooth, but turned out to be a complex, sometimes painful learning process (Stone, 2013). The focus was on redesigning the engineering processes. Whereas previously, almost all developers had worked on the same code base of the monolithic application and even minor changes required a correspondingly high level of coordination, semi-autonomous teams were formed and assigned to individual microservices. By developing standardized process models and an "internal culture of API documentation" (Lawson, 2021: 38), on the one hand, and the establishment of automated test environments and "release pipelines" (CI/CD) (Shahin et al., 2017) on the other, practices and tools were created to enable the teams to further develop the individual services of the application independently of one another and to continuously deploy their code changes. While Amazon in 2001 deployed new software components for its application twelve times a year, this number skyrocketed to 136,000 deployments per day by 2013 according to its director of developer productivity, Ken Exner (Kim, 2018: 132). The ability to continuously deploy software further enabled practices such as Canary Releases or A/B testing. Developers were able to test new functionality live with a specific subset of users and evaluate user behavior before either discarding, modifying, and retesting it or making it available to all users of the application (Kohavi et al., 2009; Feitelson et al., 2013).

At the same time, the development of software tools for the continuous integration and deployment of new functionalities served to eliminate the organizational separation of development, testing and operations in the engineering organization at Amazon (Newman, 2015). Agile teams were staffed cross-functionally and expected to take care of the entire lifecycle of the services. According to the principle "you build it, you run it" (Gray, 2006: 16), developers had to deal with the behavior of their code in operation and came into direct contact with the users of the services. This has substantially changed their work situation. As Lewis and Fowler (2014) note, regular pager alerts that got developers out of bed in the middle of the night served as "a powerful incentive to focus on quality when writing your code". The integration of development, test and operation, for which terms such as DevOps (Kim et al., 2016) and Site Reliability Engineering (Beyer et al., 2016) were later coined, is currently also being adapted in industrial companies in the context of the Internet of Things (Ziegler, 2020).

Step by step, the technological and organizational prerequisites were created to give each of these teams full business responsibility for an individual service. As companies within the company, the teams obtained far-reaching decision-making authority for the design of their services, while at the same time the performance of their services was recorded, visualized, and compared in a data-based manner, e.g., in an internally reported income statement (Stringfellow, 2018; Lawson, 2021). Performance-based monetary incentives, e.g., via stock options complemented this way of working, so that teams, as former employees described it to journalists, competed with their services in a kind of "purposeful Darwinism" (Kantor and Streitfeld, 2015).

By combining large-scale distributed systems with modularized application architectures, and the introduction of new ways of working in their engineering organizations such as DevOps, companies like Google and Amazon were able to dynamically scale complex Internet applications, handling even hundreds of millions of simultaneous accesses, keeping them highly available globally while permanently developing them further predicated on data-based observations of user behavior and operating them cost-effectively. Moreover, they transformed IT infrastructures from a scarce resource to one that tends to be abundant (Kushida et al., 2015) and created the conditions for big data processing to be used in ever broader fields of application (Grabher and König, 2020). As a result, they succeeded in building online platforms for the advertising market or retail at high speed and were able to prevail over startup competitors that were mushrooming at the same time. The network effects they were able to leverage, continuously raised the barriers to market entry for potential competitors, even to the point of establishing de facto monopoly positions in these markets.

3.3 Cloud Computing: Dynamically scalable IT infrastructures as a utility

Amazon, with its web services division, was among the first companies that systematically began tapping into the potential inherent in the independent commercialization of these basic innovations. According to Huckman et al. (2012: 4), during the modularization of the application architecture, Amazon's engineering organization experimented in mid-2002 with making information about the web shop's products available to developers of external partner companies via APIs. Partner companies were able to embed this information directly into their websites and advertise goods available on Amazon, linking these adds directly to the web store. For each sale they referred to Amazon, they received a commission of 5 to 8 percent on Amazon's sales price.

At the same time, the engineering management undertook a study, analyzing the activities for which the developer teams of Amazon's e-commerce application spent their working time. It was found that the teams devoted a lot of time to the same, comparatively repetitive activities, such as managing the computing or storage capacity required by their services. To free up the developer teams from these activities so that they could dedicate more working time to the further development of their functionalities, a set of infrastructure services was built up within Amazon (Foley, 2009). In this context, services such as Dynamo for storing objects emerged, whose APIs could be used by the e-commerce application development teams for, e.g., the shopping cart or fraud detection services (DeCandia et al., 2007).

Given the rapid success of these different measures and the observation that other startups launched similar initiatives, management reflected on the strategic importance of these experiments. In retrospect, Andy Jassy describes the deliberations as follows:

> It caused us to step back and wonder if something broader was going on. If developers would build applications from scratch using Web services, and if a broad array of Web services existed (which we believed would be the case), then the Internet would become the operating system. We asked ourselves, if the Internet became the operating system, what would the key elements be, which had already been built, and which would Amazon be best-equipped to provide for the community? At the time we were looking at it in 2003, none of the key elements of the Internet operating system had been built. When we thought about Amazon's strengths as a technology company that had simply applied its technology to the retail space first, and what Amazon had done well over the last decade, we realized we could provide a lot of the key building blocks. (cit. a. Huckman et al., 2012: 4)

According to Jassy, the results of these experiments indicated that the Internet could evolve into the operating system for more and more applications. Amazon bet big on this development. They set their sights on leveraging the core competencies they had built up while providing the e-commerce application and began to create a new business model.

Amazon subsequently worked on making individual elements of its technology platform available to external developers via the Internet. In 2006, the Web services division launched Simple Storage Service (S3) for storing objects and Elastic Compute Cloud (EC2) that enables the use of computing capacity in AWS data centers, the first two products developed genuinely for this purpose. The experience with the e-commerce application flowed into the design of these infrastructure services, which were given a scalable evolutionary architecture right from the start (Killalea, 2020: 68). The architecture of the S3 service, e.g., initially consisted of eight microservices and grew to 262 microservices by 2019 (ibid.). Customers could build their Internet applications on top of these services. Three years later, functionalities for monitoring, elastic load balancing and automatic scaling were made available, which decisively simplified the management of dynamic IT infrastructures (EC2) for user companies (Barr, 2009). The new business model was called "Infrastructure-as-a-Service".

To the extent that distributed computing, storage and network capacities, database systems and, in subsequent years, many other application components were now provided as services via an Internet connection, experience with the scalable distributed IT infrastructures could diffuse into the economy. For a fee, user companies can tap into dynamically scalable IT infrastructures to develop and operate their applications. From the perspective of the user companies, these IT resources thus become similar to utilities which, analogous to electrical energy or water, no longer must be generated, operated, and maintained in-house, but can be obtained over the Internet in near real time as required (Armbrust et al., 2009; Kushida et al., 2011). The term "cloud computing" became widespread for this model of providing IT infrastructures (Regalado, 2011). Its introduction changed the IT industry fundamentally. Back in 2005 Ray Ozzie, former chief architect at Microsoft, coined the term "Internet Services Disruption" in an internal memo to his colleagues at Microsoft, anticipating this development. It was only a few years later, however, that companies such as Microsoft, Google, Alibaba and Tencent succeeded in following suit and entered the market with their own cloud infrastructure services business.

Instead of having to invest in their own data centers, even small startups can now access the elastically scalable IT infrastructures in a public cloud for the distributed operation of their applications within a very short time and thus more quickly achieve global scale. The elasticity of the cloud infrastructures means that the development of their applications can dynamically keep pace with growth in user numbers and revenues. At the same time, they have the option of paying only for the IT resources they actually need. To the extent that this has significantly reduced the barriers to entry to building and scaling Internet applications for new startups such as Uber, Spotify or Airbnb, cloud services have acted

as “a platform for innovation and entrepreneurship” (Kushida et al., 2011: 214). At the same time, however, user companies are becoming dependent on the providers of cloud services.[15] While the production and distribution of utilities such as energy and water are highly regulated due to comparable dependencies of the user companies and the fact that they are largely publicly owned, this has hardly been the case for cloud computing so far (ibid.).

But even though access to IT resources now seems just a mouse click away, the development and operation of complex applications based on cloud infrastructures presented user companies with enormous challenges. There were no blueprints for this either. One of the pioneers in this field was the startup Netflix. Originally launched in the U.S. as an Internet application for DVD rentals, the company worked to provide a streaming application for series, movies, etc. over the Internet from around the mid-2000s, given the continuing improvements in Internet data transfer rates. The monolithic application that had been built for the Internet application for DVD rental could still be operated on a comparably manageable number of their own servers. However, due to growing demands, e.g., increasing user numbers, Netflix employees repeatedly had to deal with data loss even in this phase. It was therefore already becoming apparent that the existing IT infrastructures would be far from sufficient for the intended streaming application.

Against this backdrop, the startup conducted experiments in parallel. On the one hand, it tested a significant increase in its own data center capacity, while on the other hand it examined the possibility of using IT infrastructures from AWS. The latter option delivered significantly better results, so management made the decision to build the new streaming business model on this basis. An extensive learning process followed. Over several years, as the startup’s then cloud architect Adrian Cockroft reports, the company’s own servers were gradually switched off and use cases migrated to the cloud:

> Through 2009, we explored the cloud platform with several pathfinder projects and non-customer-facing workloads such as encoding and Hadoop-based log analysis. In early 2010 we brought up the first customer-facing workloads, starting with the simplest ones with fewest dependencies, and gradually filling in the data sources until almost everything is running in the cloud, but with the data resident in both cloud and datacenter. In 2011 we gradually moved the ‘source of truth’ systems into the cloud, with copies in the datacenter as needed. (cit. a. Farrow, 2012: 44)

To leverage the potential of scalable cloud infrastructures, the monolithic architecture of the application had to be broken up at Netflix, too. It was separated into many loosely coupled microservices, which were assigned to individual teams of developers that were to assume responsibility for their development, operation, and economic viability (Bukowski et al., 2016). During this, in the engineering organization at Netflix several software tools have been created for the development and operation of micro service-based applications on cloud infrastructures. These include, for example, “chaos engineering” technologies that randomly trigger automated failures to permanently test the reliability of the application while running in the AWS cloud (Izrailevsky and Tseitlin, 2011). Over time, their source code was made available by Netflix in open source projects to promote further development and share resources and costs with other companies.

Netflix’s rise to become one of the world’s largest streaming services, with its massive data throughput, served to demonstrate the power, reliability, and cost-effectiveness of IT infrastructures in a public cloud. As a result, not only startups but also established companies increasingly began to migrate parts of their IT infrastructures to the rapidly growing clouds of tech companies and shut down their own data centers.

4 Conclusion and Outlook: The New IT-Based Machine Systems as the Foundation of Online Platforms

The analysis of the tech company nature of online platform firms has shown that the cost-efficient provision and continuous development of platforms and other applications via the Internet, with several billion accesses worldwide every day, is by no means as trivial as it appears to the users of PCs, smartphones, or tablets. Rather, it is based on a new technological and organizational foundation, whose buildup and mastery required the development of comprehensive new capabilities and was accompanied not least by numerous computer science breakthroughs. Management and tech workers in the startups of the Internet economy could not buy these skills; they built them. In retrospect, Andy Jassy formulates this from a management perspective as follows:

> (…) most people (excluding CIOs and CTOs) didn't realize the extensive and complex technology infrastructure required to operate Amazon. You could not buy the software necessary to operate at Amazon's scale. You may have been able to buy pieces, but they would have needed to be highly customized and carefully strung together. Amazon built virtually every piece of software necessary to run a Web business that could scale, on demand, to virtually any level imaginable. Only a handful of companies around the world could claim that level of software competency. (cit. a. Huckman et al., 2012: 4)

Amid the collapse of the New Economy, new IT-based machine systems for the development, operation and scaling of Internet applications were created on the West Coast of the US. While they laid the foundation for the dynamic growth of online platforms, they have largely gone unnoticed in the ongoing debate on their conceptualization in the social sciences. However, online platform firms are completely dependent on their functioning and appeal. Downtime, outages, data loss and security breaches jeopardize their course of business.

In this novel spectrum of capabilities, three dimensions can be analytically distinguished. First, the applications are based on gigantic distributed systems of mass-produced servers which can be scaled dynamically and operated like a single computer. Second, they are founded on a modularized application architecture consisting of a multitude of loosely coupled microservices. These microservices can be developed, operated, and maintained largely independently of one another, so that Internet applications can be upgraded with new functionalities and adapted to changing market conditions not only cost-effectively but also more quickly. Third, new ways of organizing work were introduced in the engineering organizations for the development and operation of Internet applications. Semi-autonomous and cross-functional teams were formed to take on full responsibility of individual microservices over their entire life cycle. In combination, the resulting dynamically scalable IT infrastructures constitute new IT-based machine systems which are being developed and operated by tech workers. For the tech company as the inner mode of production of platform firms, they are of comparable importance as the classic machine systems are for the production processes of industrial corporations.

The buildup and mastery of the new IT-based machine systems has been crucial for turning a web shop for books into an "Everything Store" or a college network into a global social network with over two billion users.[16] After building huge clouds, they can be sourced by user companies as a utility and become a quasi-standard in more and more parts of the economy. Meanwhile, work on their development continues at breakneck speed. Like Marx's observation on the introduction of machinery in the industrial enterprises of Great Britain, "new methods of reproducing it more cheaply follow blow upon blow, and so do improvements, that

not only affect individual parts and details of the machine, but its entire build" (Marx, 1996: 408). The competencies that were built up in dealing with these new IT-based machine systems also formed the breeding ground for several other basic innovations (e.g., in the field of machine learning).

This analysis presented on the tech company nature of platform firms further demonstrates the analytical productivity of a sociological research perspective that seeks to comprehend their inner mode of production and dynamic dimensions. While studies have so far focused on the analysis of the strategy *patterns* and their evolution, this approach emphasizes the social process of their *formation* and the work of the people inside platform firms. It renders visible the complex learning processes in which tech workers have built up the technological and organizational competencies as well as the practical and experiential knowledge that made the development and implementation of online platforms and thus the monopolization of network effects possible in the first place. What, e.g., Teece et al. (1997: 518) say about the "dynamic capabilities", which they see as crucial to corporate success, seems to apply to a whole range of these competencies: "they typically must be built, because they cannot be bought." To the extent that with the rise of the Internet of Things "tech" is becoming a "layer" (Mims, 2018) in every industry, startups and established corporations, in particular, are experiencing this time and time again.

Future research on the tech company nature of platforms needs to both deepen the analysis on the genesis of the new IT-based machine systems and follow their ongoing evolution. This conceptual paper, e.g., has only been able to touch on the fact, that realizing the potential of dynamically scalable Internet applications required the complementary development of new concepts such as DevOps for organizing work, shaping work culture, and building up qualifications, particularly in the engineering organizations. But how exactly DevOps concepts and practices are implemented as well as how DevOps affects the work situation of tech workers and is experienced by them, needs to be empirically investigated from a sociology of work perspective in more detail in the future. But also, for other dimensions of tech companies the question arises as to what extent have there been developed specific capabilities and how are these implemented in practice. These dimensions might include management practices, the design of the internal organizational and governance structures, the realm of cross-company cooperation in open source projects or the process of strategy formation under conditions of extreme uncertainty.

References

Allman E (2012) Managing Technical Debt. Shortcuts that Save Money and Time Today Can Cost You Down the Road. *ACM Queue 10*(3): 1-8.

Altmann N and Bechtle G (1971) *Betriebliche Herrschaftsstruktur und industrielle Gesellschaft. Ein Ansatz zur Analyse*. München: Hanser

Armbrust M, Fox A, Griffith R, Joseph A D, Katz R H, Konwinski A, Lee G, Patterson D A, Rabkin A, Stoica I and Zaharia M (2009): Above the Clouds. A Berkeley View of Cloud Computing. Available at: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf (accessed 17 July 2021).

Baldwin C Y and Clark K B (2000) *Design Rules. The Power of Modularity*. Cambridge [MA]: MIT Press.

Barr J (2009) New Features for Amazon EC2. Elastic Load Balancing, Auto Scaling, and Amazon Cloud Watch. Available at: https://aws.amazon.com/de/blogs/aws/new-aws-load-balancing-automatic-scaling-and-cloud-monitoring-services/ (accessed 17 July 2021).

Barroso L A, Dean J and Hölzle U (2003) Web Search for a Planet. The Google Cluster Architecture. *IEEE Micro 23*(2): 2-28.

Barroso L A, Clidaras J and Hölzle U (2013) *The Datacenter as a Computer. An Introduction to the Design of Warehouse-Scale Machines*. San Rafael [CA]: Morgan & Claypool.

Beyer B, Jones C, Petoff J and Murphy N R (2016) *Site Reliability Engineering. How Google Runs Production Systems*. Sebastopol [CA]: O'Reilly.

Boes A and Kämpf T (2007) The Nexus of Informatisation and Internationalisation: A New Stage in the Internationalisation of Labour. Huws U (ed) *Defragmenting towards a Critical Understanding of the New Global Division of Labour*. London: Analytica Publications, pp.193-208.

Boes A, Gül K, Kämpf T, Langes B, Lühr T, Marrs K, Vogl E and Ziegler A (2018) *Silicon Valley. Vorreiter im digitalen Umbruch. Folgen für Deutschland und Europa*. München: ISF München.

Böhle F (1994) Relevance of Experience-based Work in Modern Processes. *AI & Society 8*(3): 207-215.

Borrus M and Zysman J (1997) Globalization with Borders. The Rise of Wintelism as the Future of Global Competition. *Industry and Innovation 4*(2): 141-166.

Brewer E (2001) Lessons from Giant-Scale Services. *IEEE Internet Computing 5*(4): 46-55.

Bukowski E, Moyles B and McGarr M (2016) How We Build Code at Netflix. Available at: https://netflixtechblog.com/how-we-build-code-at-netflix-c5d9bd727f15 (accessed 17 July 2021).

Burns B (2018) *Designing Distributed Systems. Patterns and Paradigms for Scalable, Reliable Services*. Sebastopol [CA]: O'Reilly.

Butollo F and de Paiva Lareiro P (2020) Digitale Revolution? Widersprüche der Produktivkraftentwicklung im Postwachstumskapitalismus. *Das Argument* 335: 82-102.

Butollo F and Schneidemesser L (2021) *Data and Digital Platforms in Industry. Implication for Enterprises Strategies and Governance*. Berlin: Weizenbaum Institute for the Networked Society – The German Internet Institute.

Cusumano M A, Gawer A and Yoffie D (2019) *The Business of Platforms. Strategy in the Age of Digital Competition, Innovation and Power*. New York: Harper Collins.

Dean J and Barroso L A (2013) The Tail at Scale. *Communications of the ACM 56*(2): 74-80.

DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P and Vogels W (2007) Dynamo. Amazon's Highly Available Key-value Store. *ACM SIGOPS Operating Systems Review 41*(6): 205-220.

Dorschel R (2022) Reconsidering Digital Labour. Bringing Tech Workers into the Debate. *New Technology, Work and Employment 36*(4): (forthcoming).

Evans B (2019) Netflix is Not a Tech Company. Available at: https://www.ben-evans.com/benedictevans/2019/7/31/Netflix (accessed 17 July 2021).

Evans D S and Schmalensee R L (2016) *Matchmakers. The New Economics of Multisided Platforms*. Boston: Harvard Business Review Press.

Farrow R (2012) Netflix Heads into the Clouds. Interview with Adrian Cockroft. *;login:, 37*(1): 44-46.

Feitelson D G, Frachtenberg E and Beck K L (2013) Development and Deployment at Facebook. *IEEE Computing*: *17*(4): 8-17.

Foley J (2009) Chief of the Year. Amazon's Werner Vogels. Available at: https://dsimg.ubm-us.net/envelope/259152/465273/InformationWeek_Analytics_Alerts_chiefoftheyear.pdf (accessed 17 July 2021).

Frenken K and Funfschilling L (2020) The Rise of Online Platforms and the Triumph of the Corporation. *Sociologica 14*(1): 101-113.

Fulton S (2015) What Led Amazon to its Own Microservices Architecture. Available at: https://thenewstack.io/led-amazon-microservices-architecture/ (accessed 17 July 2021).

Gawer A (2014) Bridging Differing Perspectives on Technological Platforms. Toward an Integrative Framework. *Research Policy 43*(7): 1239-1249.

Gawer A and Cusumano M A (2002) *Platform Leadership. How Intel, Microsoft, and Cisco Drive Industry Innovation*. Boston: Harvard Business School Press.

Ghemawat S, Gobioff H and Leung S-T (2003) The Google File System. *ACM SIGOPS Operating Systems Review*, *37*(5): 29-43.

Grabher D and König J (2020) Disruption, Embedded. A Polanyian Framing of the Platform Economy. *Sociologica 14*(1): 95-118.

Gray J (2006) A Conversation with Werner Vogels. *ACM Queue 4*(4): 14-22.

Greenpeace (2017) Clicking Clean. Who is Winning the Race to Build a Green Internet?. Available at: https://www.greenpeace.de/sites/www.greenpeace.de/files/publications/20170110_greenpeace_clicking_clean.pdf (accessed 17 July 2021).

Haug W F (2020) Staabs ‚Digitaler Kapitalismus'. Eine forschende Auseinandersetzung. *Das Argument* 335: 19-56.

Helland P (2016) Standing on Distributed Shoulders of Giants. *Communications of the ACM 59*(6): 58-61.

Huckman R S, Pisano G P and Kind L (2012) *Amazon Web Services*. Boston: HBS Case Study.

Isaac M and Frenkel S (2020) Facebook Is 'Just Trying to Keep the Lights On' as Traffic Soars in Pandemic. Available at: https://www.nytimes.com/2020/03/24/technology/virus-facebook-usage-traffic.html (accessed 17 July 2021).

Izrailevsky Y and Tseitlin A (2011) The Netflix Simian Army. Available at: https://netflixtechblog.com/the-netflix-simian-army-16e57fbab116 (accessed 17 July 2021).

Jamshidi P, Pohl C, Mendonça N C, Lewis J and Tilkov S (2018) Microservices. The Journey So Far and Challenges Ahead. *IEEE Software 35*(3): 24-35.

Janeway, W. H. (2018) *Doing Capitalism in the Innovation Economy. Reconfiguring the Three-Player Game between Markets, Speculators and the State*. 2nd edition. Cambridge [UK]: Cambridge University Press.

Kantor J and Streitfeld D (2015) Inside Amazon: Wrestling Big Ideas in a Bruising Workplace. Available at: https://www.nytimes.com/2015/08/16/technology/inside-amazon-wrestling-big-ideas-in-a-bruising-workplace.html (accessed 17 July 2021).

Kenney M (ed) (2000) *Understanding Silicon Valley. The Anatomy of an Entrepreneurial Region*. Stanford [CA]: Stanford Business Books.

Kenney M and Zysman J (2016) The Rise of the Plattform Economy. *Issues in Science and Technology 32*(3) 61-69.

Kenney M, Bearson D and Zysman J (2021) The Platform Economy Matures. Measuring Pervasiveness and Exploring Power. *Socio-Economic Review 19*(4): 1451-1483.

Killalea T (2008) Meet the Virts: Virtualization Technology Isn't New, but It Has Matured A Lot over the Past 30 Years. *ACM Queue 6*(1): 14-18.

Killalea T (2020) A Second Conversation with Werner Vogels. *ACM Queue 18*(5): 67-92.

Kim G, Humble J, Debois P and Willis J (2016) *The DevOps Handbook. How to Create World-Class Agility, Reliability & Security in Technology Organizations*. Portland: IT Revolution.

Kim G (2018) *The Unicorn Project. A Novel About Developers, Digital Disruption and Thriving in the Age of Data*. Portland: IT Revolution.

Kohavi R, Longbotham R, Sommerfield D, Henne R (2009) Controlled Experiments on the Web. Survey and Practical Guide. *Data Mining and Knowledge Discovery 18*(1): 140-181.

Kushida K, Murray J and Zysman J (2011) Diffusing the Cloud. Cloud Computing and Implications for Public Policy. *Journal of Industry, Competition and Trade 11*(3): 209-237.

Kushida K, Murray J and Zysman J (2015) Cloud Computing. From Scarcity to Abundance. *Journal of Industry, Competition and Trade 15*(1): 5-19.

Langley P and Leyshon A (2017) Platform Capitalism. The Intermediation and Capitalisation of Digital Economic Circulation. *Finance and Society 3*(1), 11-31.

Lawson J (2021) *Ask your developer. How to Harness the Power of Software Developers and Win in the 21st Century*. New York: Harper Business.

Levy S (2011) *In the Plex. How Google Thinks, Works and Shapes Our Lives*. New York: Simon & Schuster.

Lewis J and Fowler M (2014) Microservices. Available at: https://martinfowler.com/articles/microservices.html (accessed 17 July 2021).

Lüthje B (2001) *Standort Silicon Valley. Ökonomie und Politik der vernetzten Massenproduktion*. Frankfurt/M and New York: Campus.

Marx K (1975) Theses on Feuerbach. In: *MECW 5*. Moscow: Progress Publishers.

Marx K (1996) Capital. A Critique of Political Economy. In: *MECW 35*. Moscow: Progress Publishers.

McCullough B (2018) *How the Internet Happened. From Netscape to the iPhone*. New York: W. W. Norton & Company.

McMillan R and Metz C (2012) How Amazon Followed Google into the World of Secret Servers. Available at: https://www.wired.com/2012/11/amazon-google-secret-servers/ (accessed 17 July 2021).

Mills M P (2020) *Digital Cathedrals*. New York: Encounter Books.

Mims C (2018): Every company is now a tech company. Available at: https://www.wsj.com/articles/every-company-is-now-a-tech-company-1543901207 (accessed 17 July 2021).

Newman S (2015) *Building Microservices. Designing Fine Grained Systems*. Sebastopol [CA]: O'Reilly.

O'Grady S (2014) The Scale Imperative. Available at: https://redmonk.com/sogrady/2014/12/01/the-scale-imperative/ (accessed 17 July 2021).

O'Grady S (2015) *The Software Paradox*. Sebastopol [CA]: O'Reilly.

O'Mara M (2019) *The Code. Silicon Valley and the Remaking of America*. New York: Penguin Press.

Ozzie R (2005) The Internet Services Disruption. Available at: https://www.cnet.com/news/ozzie-memo-Internet-services-disruption/ (accessed 17 July 2021).

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: W. W. Norton & Company.

Parnas D L (1972) On the Criteria to Be Used in Decomposing Systems into Modules. *Communications of the ACM 15*(12): 1053-1058.

Peck J and Phillips R (2020) The Platform Conjuncture. *Sociologica 14*(3): 73-99.

Penenberg A L (2010) *Viral Loop. From Facebook to Twitter, how Today's Smartest Businesses Grow Themselves*. New York: Tantor Media.

Popper N and Siegel B T (2020) High-Flying Trading App Robinhood Goes Down at the Wrong Time. Available at: https://www.nytimes.com/2020/03/03/technology/trading-app-robinhood-outage.html (accessed 17 July 2021).

Regalado A (2011) Who Coined Cloud Computing? Available at: https://www.technologyreview.com/2011/10/31/257406/who-coined-cloud-computing/ (accessed 17 July 2021).

Robbins J, Krishan K, Allspaw, J and Limoncelli T (2012) Resilience Engineering: Learning to Embrace Failure. *ACM Queue 10*(9): 1-9.

Satariano A (2019) How the Internet Travels across Oceans. Available at: https://www.sfgate.com/business/article/How-the-Internet-travels-across-oceans-13704126.php (accessed 17 July 2021).

Schrape J F (2019) Open Source Projects as Incubators of Innovation. From Niche Phenomenon to Integral Part of the Industry. *Convergence 25*(3): 409-427.

Schonfeld E (2008) Amazon Web Services Goes Down, Takes Many Startup Sites with it. Available at: https://techcrunch.com/2008/02/15/amazon-web-services-goes-down-takes-many-start-up-sites-with-it/ . (accessed 17 July 2021).

Shahin M, Babar M A and Zhu L (2017) Continuous Integration, Delivery and Deployment. A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access 5*, 3909-3943.

Shapiro C and Varian H (1998) *Information Rules. A Strategic Guide to the Network Economy*. Boston: Harvard Business School Press.

Srnicek N (2017) *Platform Capitalism*. Hoboken [NJ]: John Wiley & Sons.

Staab P (2019) *Digitaler Kapitalismus. Markt und Herrschaft in der Ökonomie der Unknappheit*. Frankfurt/M: Suhrkamp.

Stallkamp M and Schotter A (2021) Platforms Without Borders? International Strategies of Digital Platform Firms. *Global Strategy Journal 11*(1), 58-80.

Stark D and Pais I (2020) Algorithmic Management in the Platform Economy. *Sociologica 14*(3), 47-72.

Stone B (2013) *The Everything Store. Jeff Bezos and the Age of Amazon*. New York: Back Bay Books.

Stringfellow W (2018) Building Product Teams. Examples from Amazon, Google, Apple, Basecamp and Fogcreek. Available at: https://laptrinhx.com/building-product-teams-examples-from-amazon-google-apple-basecamp-and-fog-creek-951573196/ (accessed 17 July 2021)

Tarnoff B and Weigel M (2020) *Voices from the Valley. Tech Workers Talk about What They Do and how They Do It*. New York: Farrar Strauss & Giroux.

Teece D J, Pisano G P and Shuen A (1997) Dynamic Capabilities and Strategic Management. *Strategic Management Journal 18*(7): 509-533.

Vise D and Malseed M (2005) *The Google Story*. New York: Bantam Dell.

Vogels W (2006) A Word on Scalability. Available at: https://engineering.salesforce.com/salesforce-is-all-about-data-d6fdf57cc8eb (accessed 17 July 2021).

Vogels W (2009) Eventually consistent. *Communications of the ACM 52*(1): 40-44.

Vogels W (2019) Modern Applications at AWS. Available at: https://www.allthingsdistributed.com/2019/08/modern-applications-at-aws.html (accessed 17 July 2021).

Vogl E (2020) Open Source-Entwicklung als Quelle von Empowerment?. In: Boes A, Gül K, Kämpf T and Lühr T (eds) *Empowerment in der agilen Arbeitswelt. Analysen, Handlungsorientierungen und Erfolgsfaktoren für eine neue Humanisierung*. Freiburg: Haufe, pp. 93-109.

Will-Zocholl M (2021) Information Space(s). In: Appel-Meulenbroek R and Danivska V (eds) *A Handbook of Theories on Designing Alignment between People and the Office Environment*. London: Routledge, pp. 82-92.

Winters T, Manshreck T and Wright H (2020) *Software Engineering at Google. Lessons Learned from Programming over Time*. Sebastopol [CA]: O'Reilly.

Yegge S (2011) Stevey's Google Platform Rant. Available at: https://gist.github.com/chitchcock/1281611. (accessed 17 July 2021).

Ziegler A (2020) *Der Aufstieg des Internet der Dinge. Wie sich Industrieunternehmen zu Tech-Unternehmen entwickeln*. Frankfurt/M and New York: Campus.

Zuboff S (2019) *The Age of Surveillance Capitalism. The Fight for a Human Future at the New Frontier of Power*. London: Profile Books.

Endnotes

[1] The political-economic framework conditions of this innovation system with its center Silicon Valley, which are favorable for such disruptive projects, have been established over several decades (Kenney, 2000; Lüthje, 2001; O'Mara, 2019).

[2] In some cases they have combined both approaches and developed into „hybrid platforms" (Cusumano et al., 2019).

[3] The dazzling prefix "tech" originally stood for "technology" and was associated with companies that provided computer hardware and software technology. The Web 2.0-founded blog TechCrunch, with its widely read coverage of the Silicon Valley startup scene, turned "tech" into a popular lifestyle term that quickly expanded to include the Internet economy in general.

[4] For further information on this research see Boes et al., 2018.

[5] For example, in the patterns and practices of platform strategies of "Wintelism" (Borrus and Zysman, 1997; Gawer and Cusumano, 2002) in the PC industry.

[6] For the analyst and venture capitalist Benedict Evans (2019), on the other hand, the crucial point in calling a company a tech company is that it has to answer "tech questions" in order to defend its market position. Accordingly, Netflix can no longer be considered a tech company because it uses "tech as a crowbar," but in the current discussions about Netflix "all of the questions that matter are TV industry questions."

[7] In their paper on software development at Facebook, Feitelson et al. (2013: 9) highlight this difference as compared to classic software companies as follows: „Traditional software products are finite by definition, with delimited scope and a predefined completion date. (…) Sites like Facebook will never be completed. The mindset is that the system will continue to be developed indefinitely."

[8] Even the employees of "seasoned" platform firms like Facebook are struggling to keep their IT infrastructures up and running in the Corona pandemic in light of new records in the use of the company's social networks (Isaac and Frenkel, 2020).

[9] According to Werner Vogels, the increasing requirements can result from a variety of developments: „larger data sets, faster update rates, more requests, more services, tighter SLAs (service level agreements), more failures, more latency challenges, more service interdependencies, more developers, more documentation, more programs, more servers, more networks, more data centers" (cit. a. Gray, 2006: 14).

[10] In the perception of the public open source development is still often associated with the free software movement from which it emanated in the 1980s. For the social sciences Schrape (2019) has highlighted that these perceptions have long since ceased to correspond to the reality of open source projects. The transition to horizontally distributed systems, in particular, promoted the embedding of open source development in corporate strategies (O'Grady, 2014). To share the burden of technology development across industry, tech companies began to use open source projects as arenas for consortial cooperation. The main contributors in leading open source projects are therefore now also "corporate developers" (Vogl, 2020: 93).

[11] Over time, some of the source code for these technologies was also made available again as open source, but at least initially to a lesser extent by companies such as Google and Amazon than by companies such as Facebook, Twitter, or LinkedIn (O'Grady, 2015: 13).

[12] In software engineering the term "technical debt" is applied to refer to the burden that is constantly accumulated through trade-off decisions and by taking "shortcuts" in the development of a solution. These shortcuts can "save money or speed up progress today at the risk of potentially costing money or slowing down progress in the (usually unclear) future" (Allman, 2012: 7).

[13] As Lewis and Fowler (2014) point out, the roots of modular design principles in software development go back to the development of object oriented programming, Unix and further (see already Parnas, 1972).

[14] The extent to which this succeeds is, according to Lewis and Fowler (2014), not least a question of implementation in practice: "That's not an absolute, some changes will change service interfaces resulting in some coordination, but the aim of a good microservice architecture is to minimize these through cohesive service boundaries and evolution mechanisms in the service contracts."

[15] On the one hand, this dependency becomes visible when cloud services are down, and the applications based on them are no longer available (see e.g. Schonfeld, 2008). On the other hand, by integrating cloud services into their applications, the startups tie themselves to the technologies of the cloud providers (Kenney et al., 2021). To counteract this "lock-in effect", technologies have been developed in open source projects which aim at least at facilitating switches between service providers and enabling "hybrid" or "multi" cloud strategies.

[16] As Penenberg (2010: 143ff) describes, at Friendster, for example, the IT infrastructure regularly collapsed for extended periods during the crucial growth phase. McCullough (2018: 261) sums up: “The engineering challenges of delivering what was quickly becoming a deluge of content were at a whole new scale, and Friendster simply wasn’t up to the challenge.”