WIRTSCHAFTSUNIVERSITÄT WIEN
Vienna University of Economics and Business

Bachelor Thesis

Titel of Bachelor Thesis (english)	The Motivation of Proprietary Software Companies to Engage with Open-Source Software Ecosystems
Titel of Bachelor Thesis (german)	Die Motivation proprietäre Software Unternehmen, sich mit Open Source Software-Ökosystemen auseinanderzusetzen
Author (last name, first name):	Gerger Konrad
Student ID number:	1027118
Degree program:	Bachelor of Business and Economics, BSc (WU)
Examiner (degree, first name, last name):	ao.Univ.Prof. Mag. Dr. Rony G. Flatscher

I hereby declare that:

1. I have written this Bachelor thesis myself, independently and without the aid of unfair or unauthor-ized resources. Whenever content has been taken directly or indirectly from other sources, this has been indicated and the source referenced.
2. This Bachelor Thesis has not been previously presented as an examination paper in this or any other form in Austria or abroad.
3. This Bachelor Thesis is identical with the thesis assessed by the examiner.
4. (only applicable if the thesis was written by more than one author): this Bachelor thesis was written together with

   The individual contributions of each writer as well as the co-written passages have been indicated.

15.11.2019
Date

Gerger Konrad
Signature

WIRTSCHAFTSUNIVERSITÄT WIEN
Vienna University of Economics and Business

The Motivation of Proprietary Software Companies to Engage with Open Source Software Ecosystems.

Institution
Vienna University of Economics and Business
Institute for Management Information Systems

Supervisor
ao.Univ.Prof. Mag. Dr. Rony G. Flatscher

Author
Konrad Gerger
H1027118
2019-09-19

Table of Contents

1 Introduction 3

1.1 Definition 4

1.1.1 Free and Open Source Software 4
1.1.2 Proprietary and FOSS Software 4
1.1.3 Shareware, Freeware and Public Domain Software 5
1.1.4 Intellectual Property, Foundations and Licenses 6
1.1.5 Relevant Open Source Terms 8

1.2 Historical Evolution of Open Source 10

1.2.1 Early Binary Era (1950) 10
1.2.2 SAP, Microsoft and BASIC (1970) 11
1.2.3 Free Software Movement and UNIX (1980) 12
1.2.4 GNU/Linux (1990) 14
1.2.5 Post Dot-Com Crisis (2000) 17
1.2.6 The Rise of the Start-Up Era (2010) 19

2 Products and Strategies of OSS Projects 21

2.1 Where to find Open Source 21

2.1.1 Mobile Devices 21
2.1.2 Infrastructure and Supercomputer 21
2.1.3 Desktop Operating Systems 22

2.2 Open Source versus Closed Source Code 22

2.3 Technological Evolution 23

2.3.1 Enterprise Resource Planning Software 26
2.3.2 Shifting from On-Premise to Cloud 27
2.3.3 ERP Market 28

2.4 Business Strategy 28

2.4.1 Dipping Market 29
2.4.2 Pull- and Push-Markets 30
2.4.3 Collaboration 30

3 Open Source at Proprietary Companies 31

3.1 Incentives and Advantages of Open Source Engagement 31

3.1.1 Innovation Capabilities 32
3.1.2 Increasing Sales 33
3.1.3 Reducing Costs 34

3.1.4 Case Study SAP 35

3.2 Possible Disadvantages of Open Source Engagement 36

3.2.1 Licenses Incompatibility 36
3.2.2 Malicious Code Contribution 36
3.2.3 Security of Retired and Unpatched Libraries 37
3.2.4 System Stability 37
3.2.5 Monetary Risks 37
3.2.6 Legal Risks 37

3.3 Engagement with the Open Source Ecosystems 38

3.3.1 Consuming Open Source 38
3.3.2 Contributing to Open Source 39
3.3.3 Owning Open Source Software 40

4 Open Source Software at SAP 42

4.1 Software Tools 44

4.1.1 Contributor License Agreement Tool 44
4.1.2 SAP Cloud Platform 45
4.1.3 Openui5 and SAPUI5 46
4.1.4 Chevrotain 48
4.1.5 SNAP! 48
4.1.6 BUILD 49

4.2 Contributions 50

4.2.1 Linux Foundation 50
4.2.2 Apache Software Foundation 53
4.2.3 Eclipse Foundation 53
4.2.4 OpenStack Foundation 54
4.2.5 Cloud Foundry Foundation 54
4.2.6 jGit 55
4.2.7 Cloud Foundry 55
4.2.8 Kubernetes 56

5 Conclusion 57

6 Glossary 60

7 References 61

1 Introduction

While there have been several studies connecting the behavior and the motivation of adopting open source software (InfM90), there is hardly any literature about the motivation of former proprietary software companies starting to engage with the open-source ecosystem. Although a collaborative way with open source comminutes was already seen as a vital source of new input, recent events have shown that open source software has gained importance around the proprietary software industry. This thesis considers different factors such as society, technology, and industry to understand the context of changes and which circumstances are influencing an adoption decision. Further research studies have primary discussed the motivation of software developers to contribute to the open source ecosystem (JosT13), whereas this study will focus on organizations as a collective group of individuals.

This bachelor thesis aims to analyze the motivation of former strictly proprietary software companies, such as SAP, to engage with open-source software. There will be a detailed view of the reasons behind such adaptations, and it will evaluate those transitions from a variety of aspects, as described above such as social, industrial and technological developments and their influences. Those aspects need to be further inspected from two dimensions: vertical and horizontal. Therefore, across different industries and eras of technological evolvements on a time axis. However, to limit the scope of the thesis, it will be focused on a narrow area of business-relevant software products and providers of ERP systems and its enabling technologies. Especially Cloud Computing and peripheries like APIs, Dockers and licensing system, which can be considered as enabling technologies.

This thesis should be supportive in achieving a better understanding of proprietary software companies and their motivation for such strategic decisions. Moreover, it should clarify which shifts and influences led to pro open source strategies.

The opening chapters will be dedicated to the definition and history of the broad area of open source software to give a solid foundation for further explanations. As well as, an overview of essential synonyms and developments.

1.1 *Definition*

In this chapter, there will be some definitions of relevant license types for this thesis, including the classification of free and open-source software and the distinction between them and proprietary software. This section will be followed by an overview of significant events of the software industry and technologically evolutions.

1.1.1 Free and Open Source Software

Open source software is a licensed software with permanent permission to study, modify, and distribute to anyone and any purpose (Open07).

The differences between free software and open source software (FOSS) is less noticeable than the difference between FOSS to proprietary software. Free software was the initial countermovement to proprietary software. In 1989, an alternative term was sought, since “free” was often mistaken for “free of charge” which was seen to harm the business side of the industry. Besides, the term “Free Software Movement” was perceived among the developer community and the software marked as moralizing and confrontational (FrWi18).

1.1.2 Proprietary and FOSS Software

Proprietary software differentiates from FOSS in many aspects. As the name already indicates, the source code is accessible by everyone, as opposed to proprietary software where the source code is restricted. With the purchase of a proprietary license, the buyer only buys specific and limited rights to use the product, making the user dependent on the issuing company. That may be problematic in several situations, such as if an error occurs and the problem is not solved, or the selling company goes out of business and stops the support of their software. Also, the implementation of restricted software might be problematic if custom-made changes are required, or further software components need to be connected.

There are also some distinct differences between proprietary and open software in respect of the development process. The former is commonly build within a restricted number of institutions and partners. The number of developers involved might be relatively small compared to open- and community-developed projects. Furthermore, proprietary software is mostly developed with a pre-defined goal to solve a given or

predicted problem. Therefore, a proprietary developed software solution will be rather programmed with specific functions for a specified targeted audience. The requirements are often laid out by sales staff, trend and market analysts, and top management to position the product to maximize profits. When creating such projects, it is common to create a schedule and a budget plan, as well as a release plan with a specific proprietary license, in order to create a commercially successful product.

By contrast, behind free and open software, there is often a whole community where potential future users are usually exposed early to the features and functions of the developed software. That process makes it possible to develop the software taking the end user's needs into consideration and reduces the chance that the software misses the target audience. The motivation behind open source projects evolves mainly out of a specific need for a solution to a problem, rather than to gain profits. That focus and empathy for a specific problem and the focused effort to overcome that with a collaborative and community-based development approach is an advantage of open software development. In order to guarantee the openness and independence of software over an uncertain amount of time, non-profit trade organizations can be established to govern such projects.

In chapter 4, there will be an overview of existing foundations and how they are placed in the open source ecosystem and how proprietary software companies can collaborate with such institutions.

1.1.3 Shareware, Freeware and Public Domain Software

It is important to distinguish open source software from shareware, freeware, or public domain software. Shareware and freeware are a kind of proprietary software distributed without a fee. However, those licenses commonly grant no access to the source code of the software, and further, restricts the possibility to modify or redistribute without the author's permission. While shareware is commonly distributed free of charge, it has often an intent to monetize indirectly. This can be, for instance, accomplished through a free personal usage, but with a pricing model for commercial usage, or the software might have limitations that can be unlocked with an upgrade. WinRAR is one example of shareware software (ShWi18).

Freeware is defined as a product free of charge and without any other fees. However, the software is categorized as proprietary software, due to the restricted access to its source code. The software functions as a black box, thus the user has no insights about underlying written code. One example of freeware is Skype or Adobe PDF Reader (FrWa18).

Public domain is a creation with no specified ownership, trademark, or patent. It can be modified, redistributed, and sold without a prior grant of permission. The author of the creation must actively mark it for public usage (SoWi18).

Open-source software is commonly created within a community of developers. Because of that approach, it has some advantages in comparison to proprietary software. However, there are also cases where open access might bear disadvantages, therefore, another way to create software might be more beneficial. The following chapters will explain some of the relevant terms and concepts to get a better understanding of some of the critical characteristics of open-source software.

1.1.4 Intellectual Property, Foundations and Licenses

A piece of software becomes open-source software once it is intentionally licensed under an open-source license. This software might be distributed for free, and it gives open access to its source code; all this happens with specific rules, which are defined by its licenses. Since the Berne Convention in 1988, every creation gets by default exclusive copyright, even without an explicit registration for such. Inherent copyright is given to the initial author of the creation itself. Therefore, it regulates the intellectual property, and it is enforced by civil law and by the *World Intellectual Property Organization,* also known as WIPO (RiIn17).

On the other hand, open-source licenses are often issued and regulated by foundations, which are organizations that oversee the usage rights and assure quality standards. By reviewing software, they can ensure that standard criteria are met and can grant an individual license. The role of foundations is much broader than just granting licenses and there is a more detailed observation in the upcoming chapters.

This section will contain a brief overview of the most used FOSS licenses in the industry:

GNU General Public License – is a widely used FOSS license and was initially written by Richard Stallman. There are currently three different versions of the licenses released. The licenses are used by software applications and operating systems like Linux or the GNU Compiler Collection (GnWi18). It is run by the Free Software Foundation (FSF) and embodies the principles of copyleft, the four freedoms of the free software movement.

Copyleft – is a form of copyright. However, it distinguishes itself by granting usage rights rather than restricting the underlying creation. It was founded as a countermovement to conventional copyright by R. Stallman (CoWi18).

Four Freedoms – is the fundamental concept of the free software movement, which has the following four pillars (GnFr96):

> *Freedom 0: The freedom to run the software as wished.*
>
> *Freedom 1: The freedom to study how a program works.*
>
> *Freedom 2: The freedom to redistribute copies of software.*
>
> *Freedom 3: The freedom to distribute copies of modified versions.*

Affero General Public Licenses *(AGPLv3)* – is a strong copyleft license based on GPLv2. The license is governed by Affero, Inc. (AfWi18). It is compatible with the GPLv3 license; however, due to the strongly restricted usage, compatibility problems might occur. Therefore, many open source projects like to avoid that license; for example, Google prohibits the usage of this license in all their open-source projects (OpTh16).

Apache Licenses *(APL)* – is a permissive and free license distributed by the Apache Software Foundation (ApcW18). The foundation has released two versions of the license since it was founded in 2000. The advantage of this license for many businesses is that it is less restrictive compared, for example, to the GNU GPL. However, it regulates all relevant areas for open source projects like the distribution,

modification, and redistribution of the underlying software for any purpose. Due to these advantages, it is the preferred open source license by companies like Google, SAP, and Microsoft.

Mozilla Public License *(MPL)* – is an open and permissive license released in 2013 by the Mozilla Foundation. It allows the indirect combination of different GNU GPL licenses and is therefore useful to integrate for larger open projects (MPLW18).

Berkeley Software Distribution *(BSD licenses)* – The BSD licenses is an open and permissive license governed by Regents of the University of California. It is in the third version and is less restrictive than other FOSS licenses such as the GNU GPL. BSD licenses are commonly used and have become a synonym for licenses with minimum restrictions (BSDW18).

Finally, several software companies started to release their open source licenses (CoFr18). They can serve those companies to prevent specific legal issues, but they can also impose barriers for developer communities to adopt or combine software with those licenses. Some examples of licenses distributed from proprietary companies are:

- *Apple Public Source Licenses*
- *IBM Public License*
- *Microsoft Public License*

1.1.5 Relevant Open Source Terms

Open Source Organizations - several organizations have formed around open-source software developments to govern the projects. For example, Eclipse, an integrated development environment (IDE), was initially founded at IBM. In 2004, it was established as a separate entity and eventually became an independent open-source foundation (EcWi18).

Cloud Foundry was initially a department at VMWare; in 2011, it became an independent foundation (ClWi18). It was beneficial for those companies to form an external and independent foundation for several reasons. One primary advantage is that the independent foundation is not bound to the usual corporate restrictions. Therefore, contributions from external parties or institutions can be managed with

substantially less bureaucracy. That enables the organizations to operate and make decisions in less time. Thereby it is not necessary that only one company forms an external organization. By establishing a neutral place, several companies can contribute to a project. Another essential aspect of such independent organizations is the neutrality of the project. Since developing a project together with another company imposes the risk that one of the parties might restrict the access to the project over time or acts in another destructive way, an external organization can reduce the risk of losing an investment.

An example is a collaboration between RackSpace Hosting and NASA, which created the OpenStack project (OpTh10). OpenStack is open-source software for the virtualization of servers in cloud computing and is mainly integrated as an IaaS solution. The jointly developed software also included risks for RackSpace, since their competitors would get access to their work. However, since the project had the potential to become a new standard for virtualizing machines and high adoption rates were predicted, RackSpace decided to find other ways to generate income with the software. It is crucial for such a platform to be widely adopted to deliver user value. Also, the project needed to reach a certain number of users to set a de facto web standard to become useful, and that was only possible if enough developers and companies would adopt it. By licensing the project under an open source license, the chances to reach the critical mass were increased. Eventually, the project was a success, and soon after, several companies and developers started to contribute to OpenStack.

Community - One critical differentiation between open source and proprietary software is the community-based development aspect. Although *open source*, not per se means *community created*, the majority of projects choose this collaborative way to develop software. A widely used tool to manage contributions to the project is Git. Git is a platform where developers can upload their creations, get feedback, and improve the software further. *Linus Torvalds* founded Git in 2005 to manage a large number of contributions to the Linux project (GiWi18). Besides managing contributions, it is also a useful tool to review errors and other components and to get user feedback.

Moreover, it makes it possible to test new extensions of a program in a separate and safe environment while the main program continues to work. Therefore, developers can test an alternative version and iterate it further before a stable and final version is integrated into the whole project. This minimizes the downtime of a program and improves the user experience.

*A **fork*** describes when the source code of an open source project is duplicated, and the copy becomes an independent project (FoWi18). Due to the allowance that free and open source licenses grants, that is, the right to modify and redistribute the source code, an enormous number of variations can arise. That makes FOSS software very versatile and adjustable so that the specific needs of customers can be addressed and solved.

1.2 *Historical Evolution of Open Source*

The *Open Source Movement* started almost as early as the computer era itself, to get a better understanding of this emerging technology and its impact, there will be an overview of historical events and its contributors.

The origins of the Free Software Movement reaches back to the 1950s, which eventually led to the *Open Source Movement* in the late ’90s of the last century. In the upcoming chapter, an overview will be provided of the previous technological evolutions and how those shaped the way current technologies are used today.

1.2.1 Early Binary Era (1950)

The University of Pennsylvania developed one of the first fully electronic computers in 1945 (EnWi18). It was funded by the US Army in 1943 and was initially created to calculate the trajectory of flying objects such as missiles and to study thermonuclear weapons. The total cost of the project was 487,000 USD, which amounts to around 6,8 million USD in 2017 (ErRa07). Once the computer was running, the press referred to it as the super-brain, and already suggested the use for academic and business purposes.

A couple of years later, the first computer for commercial usage entered the market: BINAC. Most of the early computers were established in academic areas, where it was studied and developed further.

Within this time, the first businesses started to profit from this new technology. One of the most challenging aspects of early computers was the storing of information; this problem was solved with large magnetic disks.

In 1959, IBM released the computer language *COBOL*, which stands for *Common Business-Oriented Language*. It was used in operating business applications and for solving mathematical problems (CoWi18). In the following years, IBM overtook this young industry and established itself as the market leader by providing electronic devices.

1.2.2 SAP, Microsoft and BASIC (1970)

When IBM decided to discontinue a project in the area of artificial intelligence for enterprise program software, five engineers saw an opportunity to start their own business in 1972. Eventually, in the same year, the group founded the company SAP, which stands for *Systems Analysis and Program-Development* (SaWi18). The engineers developed a fully digital system for payroll and accounting tasks, instead of mechanical punch cards, the industry-standard at that time. The system was able to process information in near real-time and introduced a substantial improvement to the previous system.

Three years later, in 1975, Bill Gates entered a partnership with IBM, and Microsoft was founded. The original software from Microsoft was an interpreter called BASIC for the Altair 8800. Both the software and the computer were famous among hobbyists. Despite the rising sales numbers of the Altair 8800 computer, Gates only sold a fraction of his BASIC software compared to the hardware units sold. Eventually, he realized that hobbyists had found a way to copy his software illegally and to avoid paying its fee. This early software was stored on a paper roll with holes. Students just made physical copies of the software without paying the fees. The software itself was sold for 500, - USD or for 75, - USD if purchased together with the hardware from IBM. As a reaction to the illegal software usage, Bill Gates wrote an open letter to the hobby community and claimed that it was justified to charge fees for software. The community did not accept his attempt to collect royalties. Consequently, further software releases had no access to the source code and restricted the unauthorized usage of the underlying software or any modification.

Some of the computer hobbyists criticized those restrictions of the source code and started a counter-movement, which will be described in the upcoming chapter.

1.2.3 Free Software Movement and UNIX (1980)

A couple of years earlier, AT&T's Bell Research Laboratories was working on a new operating system called *UNIX*. The underlying language was C, and through its high-level characteristics, adaptations with other computer systems were possible. It would later become the foundation of many other relevant operating systems like Mac OS X and Linux (UnWi18). Because of an antitrust case, AT&T was not permitted to charge any fees for its operating system, forcing the company to give a license free of charge to any interested user.

Because of the free access to the software, it soon became well-known among academic communities and businesses. Eventually, users started to report bugs, gave feedback on how to solve occurring errors, and even to create new versions of the software. In the early '80s, AT&T launched an internet platform called "Usenet" to support the communities around the operating system and to manage the contributions to the software. However, AT&T was working on a workaround for the antitrust lawsuit and decided to separate the UNIX division into smaller external companies, so-called "*Baby Bells."* Consequently, since the antitrust lawsuit did not apply to Bell's laboratory's external units anymore, the software company was able to charge a distribution fee for its operating system. Many consumers had already invested in the hardware, which was running the UNIX operating system. Therefore, many users saw themselves forced to pay a license fee or to bear the costs of switching to a new operating system.

Richard Stallman, who, at that time, was researching on the UNIX kernel at MIT, saw the changes as a threat to the developer community and created a countermovement. In order to give computer owners an alternative to the suddenly imposed fees for the UNIX operating system, he started to develop a free accessible software called GNU, the GPL licenses, and a foundation to support the free software movement.

The former MIT researcher was also seen as one of the fundamental contributors to the history of the Open-Source software industry. However, he did not count himself as part of the *Open-Source movemen*t. Instead, he prefers to refer to himself as a part

of the so-called *Free Software Movement*. Nonetheless, his work and contributions to the FOSS community and even to the software industry, in general, had a significant impact.

Richard Stallman was a former student at Harvard University, where he graduated *magna cum laude* in physics. After his studies, he started to work at MIT in the department for artificial intelligence and on several projects in that area. The university was among the first to implement the Usenet system, which was an early form of internet. Therefore, different academic institutions were able to collaborate in a new way. Stallman was an early adopter of this technology and was working with other academic groups on software projects. They began sharing test results, source codes, solutions and started to debug software errors. Among those groups, the first hacker community was formed. When the University introduced a new security system, which required every employee to use passwords, Stallman pointed out that a simple password can lead to a false sense of security and should not be trusted completely. In an attempt to demonstrate the weakness of a simple password, he back engineered a majority of his colleagues' passwords and sent them an email with their password. He asked all of them to use a simple "enter" comment instead of a password to show the disinterest to the administration and the management of the company. Almost a third of his colleagues followed his approach. The hacker community had different approaches to show their opinions, and most of the time, it was pursuing a playful and noble way of showing their advantage in programming skills. However, there were also so-called black hat hackers, who were trying to harm their victims in contrast to white hat hackers.

A few years later, MIT introduced new software systems for their offices and printers. However, one of the printer software was encountering a simple software error. In an attempt to solve the problem, Stallman noticed that the source code was restricted. As a result, Stallman was concerned about those new business practices in the software industry and soon started to explore possibilities to combat those changes. In his belief, it was needed to preserve the way of sharing and exploring software that he experienced in the academic environment. For him, open access to an underlying program was an essential right. Many companies in the industry had a different opinion on this topic at that time.

He eventually started to create an operating system, which would be open and stay open for everyone who wants to use it. The operating system should include all necessary tools which were initially provided by the UNIX operating system. He started to create a similar operating system to the original proprietary software in order to reduce switching costs and to give customers a choice. The program he created was called “GNU,” which was an acronym for “Gnu is Not Unix.” In 1984, he resigned from his position at MIT, to make sure his free software has as little influence from external institutes as possible. Since creating an operating system was an extensive project, he implemented pieces of already existing free shared software units as much as possible. One year later, large parts of GNU software were completed, and the first interested persons began to ask for copies (OpSt99).

In 1985, when more members joined to work on the project, he founded the *Free Software Foundation* (FSF). It was later registered as the official foundation and it was overseeing the contributions to the projects and the business side of the program. Stallman also launched in this year the *FSF Manifesto*. A document explaining the importance and mechanics of the *copyleft* concept and the Free Software Movement (PhGn85).

However, despite the effort on the operating system, a crucial part was still not completed: the kernel, GNU Hurd (LiGn10). The kernel is the centerpiece of every operating system and is responsible for managing the data flow between the hardware and the software components; it is compiled in the binary system. Even today, the GNU kernel project, which is known under the agronomy “*Hurd*” is not fully completed.

The next chapter explains how GNU was able to be completed without first finishing the kernel.

1.2.4 GNU/Linux (1990)

At the beginning of 1990, a fundamental technology shift took place. While the internet had already been around for more than two decades, it was only in use on university campuses, government departments, and some businesses. However, with the introduction of internet browsers and in-line images, the internet gained popularity with individual users. In 1991, Linus Torvalds made his early version for a computer kernel public. Shortly after, Richard Stallman, who still had difficulty finishing the GNU Hurd

kernel, started to collaborate with Torvalds to combine both projects. The outcome was a success, and the first GNU/Linux release was launched. Since it was licensed under a GPL with the open copyleft license, soon, different kinds of forks and implementations started.

One year later, in 1992, SAP released a new version of their proprietary business software, called R/3. The software marked a cornerstone in the history of the company and a shift in technology and their business model (SaWi18).

The industry was shifting from *mainframe* computing to a *client/server*-based approach. As they scaled their businesses, SAP noted that many companies across different industries had a demand for similar software integrations. With R/3, a package of different software components was formed and covered areas such as *Customer Relationship Management* - CRM, *Supply Chain Management* – SCM, and *Human Resource Management* – HRM. The business model shifted from a value shop to a value chain approach (JoOp10). That allowed the proprietary software company to deepen their software integration into the businesses of their customers. Resulting, in faster and better adjustable software components and improved communication systems within the businesses and the departments of their customers.

Besides the evolvement of their product line, SAP ported their business software to run on several UNIX and UNIX-like operating systems. They matched the systems with IBM's *Work-Station* product line-up, which were popularly sold hardware units at that time. Additionally, SAP further integrated their products with Oracle Server, making it possible to outsource customers' data centers.

In 1993, two prominent open-source companies were founded: Debian and Red Hat. Both distributed a Linux fork very successfully. Their business model was mainly around implementation and service agreements.

In 1994, Linux was released for the first time as a complete desktop operating system, and the community around the distribution multiplied rapidly.

Four years later, in 1998, the name *Open Source* was established. The term "*Free Software*" was often misunderstood, and it was believed that it harmed software sales, so an alternative was searched. Open-source software should have nearly the same

freedom as free software. Nonetheless, it should follow a different approach than the copyleft system in favor of businesses.

However, when Richard Stallman continued to work on free software rather than on open source software, he argued that both movements, despite their differences, must work together against the proprietary software industry (FrRi07). This idea was acknowledged by the proprietary software industry, which started to recognize the potential competition from open source licensed software. In 1998, some documents were leaked to Richard Stallman, which would later be referred to as the *Halloween Documents*, since they were published to the public around the 31st of October (HaWi18). The documents were written by a Microsoft program manager and contained detailed explanations of the market position, strength and weakness analysis of open source contributions, and how they could be a potential threat against the established proprietary company.

Apple and Netscape also saw the potential threat and released their open-source licenses: APSL and NPL. However, since the licenses were incompatible with many already existing FOSS licenses, especially the GPL, many developer communities saw them instead as marketing tools rather than real open collaborations and received harsh criticism (ApRi07).

However, the Open Source Movement grew steadily and received more attention as an alternative with a broad spectrum of supporters. In 1999, the Open Source Initiative received a historic latter: A company was asking the OSI community to certify their source code and declare it officially as open source, to avoid a situation like encountered it Apple with their APSL release. Many companies followed that approach and asked for certification from the organization.

In the same year, Google and VMWare were established. One year later, Salesforce was founded, which will later become a strong opponent in the CRM market for SAP.

At that time, many internet-based businesses made their companies public and went to the stock market to raise funds. In addition to the possibilities that new technology made available, the US signed off the *Taxpayer Relive Act of 1997* (DoWi18). The act reduced the taxes on the upper margins of capital gains and fueled the stock market

further. Many companies were soon after well-funded, and stockholders had high expectations. However, software companies were following a very aggressive growth business model and tended to over-evaluate their expected sales numbers.

In contrast to traditional business concepts where companies invested in physical machines and assets, many of the "Dot-Com" companies spent nearly their entire funds on marketing. Additionally, they often distributed their products for free or at a high discount in order to gain market share. Besides that, governments were investing significant funds into telecommunication infrastructure and technologies. However, the expansion trend turned when Alan Greenspan, at this time, Chairman of the *US Federal Reserve*, announced a rise in interest rates; stakeholders started to ask for earnings to cover their interests at their banks. Soon many investors began to realize that their investments and trust companies had cash flow problems and that those investments were too optimistically evaluated.

In the first half of the year 2000, many investors needed to sell their stock before Tax Day to cover their taxes for the previous year gains. When Microsoft was charged guilty of monopolization, the market started to dip and resulted in a 25% decline in the NASDAQ market in April 2000, and in just six months the internet marked plunged and bottomed out at a 75% decrease, forcing many companies into bankruptcy. After this event, which is also known as the burst of the *Dot-com bubble,* the industry needed to define how internet products and services are evaluated more sustainably.

1.2.5 Post Dot-Com Crisis (2000)

The recent internet industry collapse shadowed the early 2000s. Despite the crisis in the stock markets SAP, Amazon and eBay were experiencing positive sales numbers. Microsoft and SAP were deepening their partnership and started to increase their product integration, such as Microsoft SQL Server. That server was a direct competitor of Oracle, and almost half of SAPs customers were running their business application on Oracle servers. However, Oracle and SAP continued their business cooperation.

Nonetheless, in 2004, Oracle started to buy EPR business application provider companies systemically. Those acquisitions positioned Oracle as a direct competitor of SAP. Eventually, a fierce competition started, and several lawsuits were filed in the upcoming years (OrWi18).

In 2004, the same year that Facebook entered the market, SAP launched its new product: *SAP ERP Central Component*. The software is based on its successor R/3 and is suited for a range of different lines of businesses (LOBs). Once again, the introduction of the new product marks a cornerstone of technological advances in the business software environment and the strategy of SAP (SAWi18). Since the further integration of software-supported elements at all sorts of business-relevant areas was strengthened, more fluid communication between different departments was realized.

Regarding the business model, SAP changed to a more holistic view of software integration. The company started to consider different parties, also referred to as a "value network" business model (VaWi18). It differentiates between internal and external stakeholders, the communication streams, and if the tangible or intangible information is passed on.

However, SAP's sales numbers were mainly coming from traditional ERP systems. In 2005, Alex Atzberger, a Harvard student and later SAP employee, pointed out in a university thesis that SAP is overlooking the transformation to cloud computing and the emerging CRM market with new competition from Salesforce (AtBu16).

In 2006, Twitter entered the market, and Amazon launched their Amazon Web Service (AWS). So far, the cloud computing industry was mainly dominated by Oracle, Microsoft, and IBM; nonetheless, new competitors were entering fast, and the market was about to change substantially in the following years.

In 2007, GitHub entered the market and steadily became a de facto standard for open and shared software projects. In the same year, Oracle bought SUN Microsystem for 7 billion US dollars (OrWi18). SUN was a large hardware manufacturer and software developer, and one of the largest UNIX/Linux operating systems distributor. Soon after, the software provider also acquired MySQL. The takeover was concerning for the open-source community, and in fact, Oracle discontinued a range of products from SUN's product line, including OpenSolaris and StarOffice (SuWi18).

In 2010, SAP introduced SAP HANA: which is an in-memory cloud platform and marked the next decade for the company and the transformation from ERP to cloud computing.

In 2010, SAP bought Sybase, a company specialized in data servers, information management, and mobile data usage (SaHi18) in order to step into the server market.

In 2012, and the following years SAP increased its focus on cloud computing technologies and bought several cloud-based companies. Two of the acquired companies were Ariba, an online procurements system, and Concur, an online expense management solution. Both purchases were strategically made to enter the cloud market and extend the portfolio for their customers. In the same year, SAP launched a new product: SAP HANA Cloud Platform, which is a Platform as a Service (PaaS) for business applications.

Also, in 2012, SAP moved their entire SAP Business Suite product line to SAP HANA and was starting the transformation to the cloud-based area. The board recognized the importance of cloud computing and saw that customers want to integrate and eventually buy a variety of online applications. Primarily because of the rise of the internet and the implementation of online applications, the industries started to accelerate.

1.2.6 The Rise of the Start-Up Era (2010)

Theano, an open-source machine learning library, was started on GitHub in 2011.

In 2014, Satya Nadella became the CEO of Microsoft and changed fundamentally how the company interacts within the open-source ecosystem. In the same year, Cloud Foundry was announced to be open and governed from an independent foundation.

In 2015, Google released TensorFlow under an Apache 2.0 License. The program is an algorithm library that supports a range of AI-applications and provides deep learning mechanics. The project was initially started in 2011 at Google Brain as a proprietary machine-learning library (TeWi18). It helps with the automation of software components, enables image and speech recognition, and is becoming a core innovation for many software features of Google's product line, especially its search engine.

In the same year, Microsoft opened its machine learning library, Cognitive Toolkit, to the public. It is part of Microsoft's Cortana speech assistant and Skype.

Likewise, in 2015, Elon Musk founded a non-profit AI research initiative, OpenAI. The project aims to fight the concentration of knowledge in the area of AI/ML within a few companies and to even-out competitive advantages (OpAi18). Musk also stated his concerns for uncontrolled artificial intelligence and the potential threat to humankind publicly.

One year later, Amazon opened its machine learning library to the public. When compared to Google's library, it instead focuses on product recommendation rather than on speech recognition (AmWi16).

SAP Cloud Platform, a Platform-as-a-Service (PaaS), was released in 2016. The platform is built on open source technologies and developed with SUSE.

In 2017, the Montreal Institute for Learning Algorithms eventually released a stable software version of their open-source machine learning library Theano. The library became fast a popular software due to its free and open access, and the focus on academics graded us for mathematical expressions.

2 Products and Strategies of OSS Projects

In this chapter, is an overview of areas where OSS plays a significant role and how that might have changed over the years. There will also be a reflection on why FOSS and the proprietary software industry co-existed in parallel and why in the last two decades, the conversation about the adaptation of open source shifted. There will also be an explanation of different strategical attempts and how business models might have changed regarding technical evolutions.

2.1 *Where to find Open Source*

The free software movement was, in many aspects, a countermovement to the development of the proprietary software industry. Besides the community and collaborations aspects, a strong idealistic ideology was increasingly influencing the decisions of the free software movement. The open-source movement was formed to distinguish from the FSM, and to have a more pragmatic approach and to appeal to business aspects. On the other hand, the increase in popularity among commercial FOSS software represented a potential competition for proprietary software companies. Despite the fierce competition, open-source software started to have a vast influence on many information technology areas.

2.1.1 Mobile Devices

The global smartphone production, as part of the manufacturing industry, represents a multi-trillion USD global market (WMU18). Around 1,5 billion smartphone units were shipped worldwide in 2016 (MoOp18). The market was dominated by the Android operating system, with 87.5% in 2016. Resulting in the majority of globally used mobile devices running on open-source software, which is licensed under APL 2.0, and GNU GPL 2.0 (AnWi18).

2.1.2 Infrastructure and Supercomputer

Another area that is primarily dominated by open-source software in the field of supercomputers and infrastructure, such as severs. The top 500 supercomputers use Linux exclusively as an operating system (LinW18). Large publicly traded companies such as Amazon opted to build their software on Linux distributions like their cloud

server product EC2. Another example is the web-server market, which is mostly dominated by Linux distributions like Apache HTTP Server and NGINX (OSOp16).

2.1.3 Desktop Operating Systems

Despite operating systems for all kinds of devices, as a private desktop operating system, only 1,6% of users chose to install Linux on their private machines in 2018 (LiSt18). Similar to those numbers, products like LibreOffice, OpenOffice, VLC Media Player, or Firefox have a relatively lower adoption rate compared to other commercial products. It makes evident that the market dynamics for *business to consumer* markets (B2C) differ from business to business markets (B2B). The reasons for such differences can be based on the different approaches taken, like marketing efforts, consumer behavior, and technical support. In the upcoming chapter, there will be a more detailed explanation of the possible advantages and disadvantages of open versus closed source software.

2.2 *Open Source versus Closed Source Code*

The success of open-source software in the areas mentioned above can be traced back to a set of reasons. In general, because of the way how the software is created and how open licenses work, it follows a more inclusive approach. This results in higher adoption of open standards and therefore increases the versatility of the software. Another factor is the independence of any private or public traded software provider. The asymmetric control by a single vendor can impose a set of risks for the software consumer. Those are important factors when it comes to a variety of software users like government-owned programs. Such programs might be systems, which are used for weather forecasts, earthquake predictions, genome sequencing, or nuclear warhead simulations (SuTh18). In such matters, to rely on a single vendor and closed source code would introduce a weakness. Moreover, those programs might deal with or generate sensitive data. To minimize the possibilities of data leaks and to ensure high reliability of generated information, open source-based software can help achieve some of those requirements.

Despite the advantages of open source developed software, there are drawbacks to consider. Resulting from the high adjustability from OSS, open software often needs extensive additional adjustments and customizations. Depending on the tech-

savviness of a potential software consumer, proprietary software might be more appealing. Specifically, proprietary software is, in general, more consumer-need oriented; help for implementation, training, and maintenance are often available as side products. Proprietary companies created profitable business solutions around their products to generate secondary income streams. Most of the time, they actively consider customer needs in order to retain them and are more likely to act actively to adjust to those needs. However, that can result in an asymmetrical dependence and lead to a vendor lock-in, which is another possible disadvantage of closed source software. Because of that dependence, proprietary software provider tends to charge higher prices for their services and products.

2.3 *Technological Evolution*

It used to be that the consideration of open source products for business software such as accounting and human resource programs was primarily based on factors like total-cost-of-ownership, maintenance, support, security, and usability of the software.

Nowadays, the conversation is increasingly based upon features, compatibility, and which network effects can be realized due to a possible higher adaptation of specific software standards.

Those changes can be based on a variety of influences such as technological evolutions, change in consumer behavior, and new market strategies of the software providing companies.

Don Tapscott and Anthony Williams explained in their book, *Wikinomics*, how the industry and society, in general, have changed through the decreased communication costs in the last two decades caused by the internet (WiAn06). The idea is based on *scaling economics;* by releasing open and free accessible products, the number of users can be increased and therefore is stimulating markets around their products and growing the overall market size. Some of the companies that gave free access to products and knowledge found themselves to be very successful.

In conclusion to their studies represents an open access strategy, not necessarily an altruistic move; instead, it is good business acumen.

Hal Varian expressed a comparable market behavior in a formula and discussed these dynamics in his book, *Information Rules* (HalH98):

Reward = (Total value added to the industry) * (Our share of industry value)

Google, Microsoft, and IBM are known for following a similar strategy with their AI and ML libraries TensorFlow, DSSTNE, and many more (TeGo17). The motivation behind opening years of research and possible competitive advantage can provide a variety of advantages: First, if the industry gets stimulated due to new technologies, then market growth can be accelerated. Yet another factor is that open collaboration attracts more extensive access to experts, especially in the academic environment.

On the other hand, many similar open-source projects are emerging fast and gathering large communities like Torch, Theano, or Keras. These are machine learning libraries, which use open source licenses and are widely acknowledged within academic research institutes. Thus, to be successful in that area, proprietary companies are urged to follow a new strategy (AiGo18).

In order to establish a new machine learning library successfully, it is essential to appeal not just to businesses, but also academic communities and young developers.

Google eventually released TensorFlow, licensed with Apache 2.0 in 2015. TensorFlow is a library with mathematical algorithms also used for machine learning, such as neural networks and deep learning. It was internally developed as part of the Google Brain team, which researches in the area of artificial intelligence. Moreover, the ML-library is well documented, and Google provides courses and learning material for free to students and businesses.

Another trend which is in favor of open source is that applications and software in general, are becoming increasingly Internet-based rather than on-premises. Services are consumed on a variety of devices, which causes the whole software industry to shift to a more mobile-friendly environment. Experts are referring to this as the *computing power utility* (WaIn08). This is similar to the constant supply of water, electricity, oil, and gas, which can be consumed at any time and where the consumed amounts are what is billed. The establishment of infrastructure to distribute those utilities was a fundamental factor for the previous industrial revolution. The

commoditized power of the internet is providing businesses with high-end and state of the art computing power on a per-use cost model. Therefore, companies can implement a more extensive range of software products based on scalable cost models and at the same time, reduce the risk of opportunity costs. Simultaneously, there is an increasing offer of web-based applications and microservices which are providing features like machine learning, speech recognition, and image recognition for companies that are not specialized in any of those areas.

Those services require different approaches like orchestrating APIs, accessing several thousand servers, and virtual machines to reach computing power comparable to a supercomputer (HCP). A collective of companies are actively and passively involved, and many different technical approaches are combined, to achieve such results.

Since many applications need to cross-communicate to achieve features like those listed above, standards and open communities are becoming increasingly important. Initiatives like OpenAPI, Kubernetes, and OpenStack are providing places for companies to engage in those developments and to influence decisions about possible new standards.

Open source organizations are representing a vital solution to co-create complex programs. Consequently, open access is pivotal to establishing new software and increasing the adoption rate. Platforms like OpenStack or OpenAPI are becoming more valuable to each involved member. The more users are in the network.

Several proprietary companies are trying to profit from the increased popularity and faster adoption rate of open-sourced software. For example, there are free courses about open source languages issued by Apple to give access to their software language, Swift. Google tries to accelerate the machine learning market to open source its TensorFlow library, as described above.

Technological evolutions do not solely drive those changes. Human and social aspects are evenly influencing the changes in the software market and how open source is embraced by former proprietary software companies. For example, Satya Nadella, CEO of Microsoft and former SUN employee, is profoundly influencing how the company is changing its strategy with open-source software. The same goes for CEOs

like Jeff Bezos from Amazon or Sundar Pichai from Google, who are embracing open source and supporting the ecosystem.

2.3.1 Enterprise Resource Planning Software

ERP stands for Enterprise Resource Planning and describes in general systems that support business processes with software and technology. Those systems are used in supporting areas such as distribution chains, customer relation management, production planning, as well as human resource and accounting (EnWi18). Those systems collect, store, and interpret a broad set of information throughout all connected business units.

The German software producer SAP saw early on the need for such enterprise support and was influential in establishing the ERP market as it is known today. One of their products, named R/3, is the third iteration of their “real-time” data processing software and was released in 1992 (R3Wi18). With R/3, SAP introduced a 3-level tier system which is differing between three layers: *presentation, application,* and *database layer.*

Together with this structural separation, a client/server structure was established. This structure enabled single source and cantered storage systems for individual data sets, which used to be distributed across different systems within a company. Furthermore, it can be accessed from different access points within a corporation and externally connected partner companies.

An example of such a system integration would be if an employee is submitting a request for holidays, the system communicates with the HR software, which corresponds with the finance software, which processes the right paycheck automatically and informs managers if rescheduling is necessary. Eventually, an external catering company would be notified that one lunch meal less is required for a given period. Those integrated process can be implemented with ERP systems; they helped business in the last century to experience a production increase and to focus on their core competencies.

Around 2012, it became popular for software providers to start offering products also as *mobile phone applications,* for example, Concur. The expense management software, which was acquired by SAP in 2014, handles travel expenses of employees

and functions as a trip booking software. The system can be accessed via a web browser or a smartphone application and communicates to the backend of the intergraded ERP system.

However, those mobile software applications are rather handled as an external component of the business software system. Although most of them are well-integrated solutions, the externality introduced a variety of drawbacks, including increased response time from applications, and separation of data sets across the attached and central system, resulting in cumbersome user experience. The challenge of the following decade was to integrate all those extensions seamlessly on one platform and to reduce data silos and create a compiled access point for personal information, which could still run across different platforms and devices (AbOp18, 2018).

2.3.2 Shifting from On-Premise to Cloud

Traditionally ERP programs are hosted on local servers, also referred to as *on-premise*. On-premise hosted programs have a set of advantages and disadvantages. Though the migration from locally hosted programs to cloud solutions is a complex task, business software providers are nonetheless pushing towards the shift to cloud-hosted programs.

On-premise hosted programs offer a better individualization of the program and greater control over the data. In general, there are high one-time payments to install the software and often require additional hardware to set up the system.

Cloud computing describes an aggregation of remotely hosted computing resources utilized by a client. The client can be a website, mobile application, or desktop program, or used for storage and computing power (WhOp16).

“Cloud computing is shared pools of configurable computer system resources and higher-level services that can be rapidly provisioned with minimal management effort, often over the Internet. Cloud computing relies on sharing of resources to achieve coherence and economies of scale, similar to a public utility.” (CloW18)

In order to run programs on a cloud server, it needs to have an operating system, library, and the application. All those components are installed on "virtual machines". The virtualization makes it possible to run the application on any environment that

supports those programs, whether it is hosted online or on a physical machine. This results in an application that is more portable and enabled to run in an environment which supports that proses.

The advantages of cloud-hosted over the on-premises hosted programs can be a shorter implementation time and faster software updated from the vendor, resulting in a more reliable system. Mostly, it does not rely on additional hardware, and rather than a one-time payment, it is offered as a service and is sold as a monthly subscription. (AdER15)

2.3.3 ERP Market

The proprietary ERP market represents a large area of the software industry and is dominated by a small number of software providers. However, a large portion of the ERP market demand is met with second and third-tier software providers (ApTo17).

Open-source ERP, on the other hand, is concentrated among three market segments (UnSc15). The first segment represents companies, which have the required knowledge and skills within their company to implement an open-source system. Since this can represent a complicated process, many businesses with insufficient or little knowledge in software implementation are less likely to integrate an open ERP system successfully.

The second segment is small and mid-sized companies (SME). One of the most outstanding arguments for this group is to save on license fees and to have a flexible system. However, they might need to make a bigger one-time investment upfront and need to compare the total cost of ownership over the life period of the integration.

The last segment is the governments and research institutions. This group can be classified with their requirement of a completely open system due to their restrictions and specific needs.

2.4 *Business Strategy*

Over the last couple of decades, a variety of business strategies have evolved. In this area, proprietary and non-proprietary software companies tend to have different approaches. However, it can be observed that former strictly proprietary software

companies start to use open source components as a go-to-market strategy and to build a community around their products in order to generate traction for their brand.

Open Core describes a functioning software that is open source and freely available. Around the center is a palette of add-ons commercially available, which extends the software by various features and functions. Those offers often come in bundles together with services and help to enrichen its value. This strategy lowers the barrier to try out the software and is common around open source software businesses

Close Core, on the other hand, is practiced rather by proprietary software companies that have a dominate selling product. It is common to open source a fully functioning add-on component. By making the additional product freely available, those add-ons are more attractive to a broader audience and generate visibility for the core product.

Cloud-hosted services are built with open source code and can be made available as platform-as-a-service. They generate income by selling additional components, such as the ability to monitor dashboards and integration features, as well as other services and support.

Vendor lock-in, which is a common practice in the software and, specifically, the proprietary EPR market (VenWi18). Customers are lured into a system by buying one component and then adding on additional products and building up switching costs. However, consumer expectations and the speed of innovation have changed drastically over the last two decades. It is becoming more common that customers are open to paying as they use and try out new products if they fit their needs.

2.4.1 Dipping Market

Dipping markets or Winner-takes-it-all Markets are defined by an unequal distribution of customers, where the product with a slight advantage over the competing products gets most of the customers (WinWi18). These apply to markets like the sports industry, certain design markets or search engines, and are increasingly experienced in the software industry. This implies that dipping markets make it challenging to calculate marketing budgets or other sorts of resources since the gain from reaching the first spot is exponentially higher. On the other hand, the risk for such markets is that if the product fails, all the investments made might be worthless. Therefore, it is becoming

increasingly important for software companies to penetrate a market in a short time with their product in order to be successful, which explains the interest of closed source companies to lower the barriers to access their products

2.4.2 Pull- and Push-Markets

A push strategy implements active communication with the end customer to promote the product. It is often applied if a new product is launched or in markets that are not transparent. In contrast, pull marketing represents a situation where a product or brand is so desirable that customers will seek the product themselves and possibly buy it.

2.4.3 Collaboration

In some cases, open source is a strategy to get several companies working on one project, like in the case of OpenStack, as described in the first chapter. By registering the software under an Open Source license, legal questions regarding the ownership can be addressed. It also serves as a kind of guarantee that it will be available for one of the creators for an extended period of time. On top of that, the project can be shared among other communities such as universities or other entities of the companies without the risk of interfering with internal or external company policies.

3 Open Source at Proprietary Companies

In this chapter, it will be examined why former strictly proprietary software companies are starting to consider distributing open source software. There will be a separation between possible motivation and risks, and there will be a close observation of different ways a software company can engage with the open source ecosystem.

3.1 *Incentives and Advantages of Open Source Engagement*

There is a wide variety of publications that review the motivational background of individual developers to engage with open source software. It ranges from a general perspective on intrinsic motivation, discussed by *Edward L. Deci* in his book "*Intrinsic Motivation and Determination in Human Behavior*," to extrinsic motivation like status, career improvements and financial aspects of individuals, explained by *Krishnamurthy* in 2006 and general motivation from *Lerner* and *Triole* in 2002.

However, in this thesis, the focus will be on the motivation of a company to engage with open source software, where a company is defined as a legal entity that engages in business and is a collective of different individuals, which serve different interest groups of stockholders.

A company's prime objective is to generate value and to persist over time. All activities that support that aim can be classified as goal-enabling and are of increasing importance as the market tends to get saturated with growing competition.

It can be distinguished between three arguments, which are possible drivers behind the motivation to engage with open source software: One is reducing costs, second to gain innovating abilities, and lastly, to increase sales (WhIM12). However, conducted studies revealed that those categorizations might miss one argument: moral obligation. All companies included in the studies stated that they want to give back to the FOSS community, which is an essential indicator. It signals that the company is willing to reinvest, and that can build trust.

3.1.1 Innovation Capabilities

It is argued that if open source software has an advantage over proprietary software, it is the greater ability to innovate (HipMI05). Due to effects such as Linus's Law and access to a broader community, many studies have pointed out the positive effects on innovation (LiWi18).

Software Standards are defined as terms, concepts, formats, or styles of documentation, which are widely recognized and commonly agreed on by software creators in order to ensure quality standards and to understand other products. One definition for standards by ISO, an international institution for standardization, is: "A document that provides requirements, specifications guidelines or characteristics that can be used consistently to ensure that materials, products, processes, and services are fit for their purpose." (IsoIs18) Standards are voluntary, which means that they can be seen as guidelines, but not everybody needs to adopt them. When Apple excluded the 3,5mm audio port from their distributed phones, they changed from the standard for audio to their proprietary lightning connector. That promotes wireless headphones and sales for their proprietary lightning dongle.

Closed Standards can be concealed to the public, including the documentation and specifications. Those standards can be distributed according to specific terms and conditions and might come with costs such as license fees.

Open standards are seen to be increasingly essential to increase efficient communication about the expanding number of devices and applications. The Open Source Initiative believes that open standards should embrace: no internal secrets, availability, royalty-free patents, no NDA agreements, and no OSR-Incompatible Dependencies (OsiO18).

The active involvement in the development of a new open standard can be beneficial for commercial companies due to various reasons. On the one hand, they can be implemented early in new products. On the other hand, the involvement also allows businesses to contribute and possibly form the new standard. Upon consideration, that may explain why companies become an early part of different open initiatives like OpenAPI (OpeWi18).

3.1.2 Increasing Sales

Commercial companies are, in general, motivated to increase sales numbers. This can be done through direct marketing activities; however, other activates can also contribute to sales numbers and generate income.

Secondary products, as described in the previous chapter about software strategy, in an open core, can increase the sale of additional commercially available software components. Proprietary software companies, per contra, will instead have a closed core and distribute open software as an add-on. Such a practice can have a positive effect on the core product due to its higher agility and broader feature set.

Compliment services are widely offered in the software industry and represent a valid secondary income source. They can be offered in the form of training, certificates, technical support, or consulting services (FitzT06).

Thought leadership can be attained by active contributions to a software community, and it signals a high level of knowledge and applied skills. A company can give panel talks, participate in a conference, and host meetups. That can have a positive impact on brand awareness and attract more stockholders; moreover, it can attract future hires.

Time to market, also called TTM, refers to time spent between the idea to the final product release. There are no official standards for this measurement, but it is an indicator for industries to compare costs and product cycles. Open-source software components can have a positive impact on the TTM due to the reuse of code and accessing feedback from a broader community.

Another effect of open-sourcing source code can be the *reduced friction* to consume and to try the product. In an increasingly saturated market, it is of importance to penetrate the market with software products in a short amount of time to gain profits. Due to the maximizing effects on the zero acquisition costs, proprietary software companies can use that to their advantage.

3.1.3 Reducing Costs

Cost reduction due to the usage of Free and Open Source software might seem like a valid argument for its utilization. Software development can be accomplished faster due to the use of freely available software. However, those advantages must be held accountable when considering the overall cost of software, also referred to as total-cost-of-ownership. Since free and open-sourced software code can include broken or malicious parts, it is required to review possible implementations carefully. That can expand the expected development length of a project considerably. Another factor to be considered with FOSS is that it is possible that documentation might not be up to date, incomplete, or missing. Those are factors that need to be considered before the code is reused. In an upcoming chapter, there will be a more detailed observation of which factors of an open-source software implementation need to be considered and how those might affect the final product regarding its distribution.

The technologies and the industry are changing rapidly regarding technological evolutions, but also to how technology is consumed and how technology is taught and learned. It became a standard that young developers use platforms such as GitHub and other networks to grow their experience and reputation. Senior developers prefer working environments where they can use such platforms. On the other hand, companies are interested in keeping the training time for new employees is as low as possible to reduce costs. Therefore, it is increasingly attractive for employers to provide such tools which are widely used already.

Proactive saving by contributing to the early stages of open source projects can be another aspect of why a proprietary software company might be interested in engaging with a non-proprietary software company. For example: when consuming open-source software, the code might fit to a high degree in the already existing system and products of a company. However, the last part of the implementation might require reconfiguring a large part of the software. Also, that task can be only accomplished with the necessary technical knowledge and resources. If a commercial company can actively contribute to an open-source project of interest, it might be able to influence the product and can start early to consider how to implement the free source code. That may result in saving time and resources. Moreover, by contributing to an open-source project, the company can gather feedback from the developer community. That

feedback can reveal new methods to solve problems, which ensures that the quality of the code can be further improved, as opposed to purely in-house development.

Hiring talent can represent a challenging task for a software company; it is seen as a critical must in order to stay competitive in the market over an extended period. Access to an active community can help bridge the labor gap and get highly qualified support for projects.

3.1.4 Case Study SAP

When SAP developed its proprietary high-level programming language *ABAP* in 1983 (AbWi18), it represented a competitive advantage for the company. It was developed to fit their current needs and their entire product line. Moreover, due to their implemented restrictions, it was possible to charge higher margins, not only that, additional services could be sold, such as training and support packages.

The proprietary character of the language also represented some drawbacks. The number of developers who used the language was very limited, and due to the restricted usage, it was less attractive to young professionals.

Moreover, due to the less restrictive character of freely available and open languages such as Java, PHP, or Python, they attracted more developers in the first place. Secondly, due to the amount of available research, institutes and academic environments had more interested in improving and developing them further.

However, ABAP is still a language of significant importance for SAP and their customers, given the large number of applications that are built on it. Nevertheless, it is becoming difficult to hire talent for this proprietary language. In late 2018, SAP released a runtime environment for their cloud platform offered as a platform as a service, offering ABAP developers the ability to run programs in the Cloud Foundry Environment of the Sap Cloud Platform; giving customers the chance to cloud-enable their ABAP based programs and connect them via remote APIs to a broad variety of cloud applications. The language supports an Eclipse environment and enables the importing and managing of development statues via Git, offering access to other open-source projects (AbSa18).

3.2 *Possible Disadvantages of Open Source Engagement*

As the previous chapter shows, an open-source engagement can have a positive impact on commercial software companies and might contribute to various strategic business plans. Nonetheless, the usage of open source code might bear some disadvantages and should be considered.

3.2.1 Licenses Incompatibility

In general, software code and specific open-source code are primarily defined by its license. Over the years, a large number of software licenses were created. They each have their own characteristics, therefore, serving different purposes and coming with a different set on requirements. In general, due to copyright laws, once a part of the free or open-source code is implemented into a project, it passes on its usage right with its possible limitations. Hence, some combinations of licenses work in favor of each other and permit each other to work in parallel. On the contrary, it might occur that two licenses contradict each other and may cause conflicts. Moreover, it can become challenging to create a proprietary software product once open-source licenses are included since it might not be permitted to sell or restrict the usage rights.

Besides license incompatibilities, there are also compliance restrictions due to antitrust laws. In this sense, once an external developer contributes to an open project, the submitted code might be copyrighted. Additionally, in the case that the developer is employed, the contributions might be owned by the software company. Therefore, it is, in some cases, essential to file agreements with all contributors to try to avoid legal actions. There will be a more in-depth review of CLAs in an upcoming chapter, which will point out different kinds of agreements and in which cases they are applicable.

3.2.2 Malicious Code Contribution

Due to the accessibility of an open system, it can also be a source of security vulnerability. Malicious code, leaking areas, or gateways might be introduced and can cause severe problems. It is argued that open-source software, in general, might be more secure, since it is tested and reviewed by a wide variety of software experts with different aspects. However, a planned security gateway can be implemented and used deconstructivity, if the necessary security reviews have not proceeded and access rights are correctly managed.

3.2.3 Security of Retired and Unpatched Libraries

Open-source software is often managed by a voluntary gathering of developers connected remotely; software projects can get outdated once the community ceases to exist or becomes smaller. As a result, new security updates might not be implemented, and libraries might become unpatched. That is problematic if the source code is implemented without a thorough and careful compliance review.

3.2.4 System Stability

There is always a chance that implemented code causes conflicts with the existing code. In order to avoid more significant losses, a regular building plan and implemented automated tests are required to avoid more significant problems along the way of the project. If a problematic area is not spotted early o, it might cause a crash of the program, and back-engineering might become a resourceful task.

3.2.5 Monetary Risks

Projects naturally bear the risk of failing and might fail to monetize successfully if this is important for a project's sustainability. The risk is over-proportionally higher if a long development cycle is in place, with possibly several hundred developers, and in some cases, an innovative product. There is always a chance of non-acceptance by the market; features of the product might become outdated or become faced with stiff competition, thus impeding growth. Nevertheless, open-source and community-based approaches are ideal to bootstrap a project. This can have a positive effect on keeping costs down and expenditures very low until the project starts to sustain itself.

3.2.6 Legal Risks

Open source is based on licenses with foundations in place to govern those licenses and hold companies and developers accountable for applying them correctly; consequently, there will always be a legal risk. It is good to be proactive and become involved early in order to manage licenses correctly and state user and contributor rights, from the beginning of a project according to legislation. However, there is always be a chance that certain rights be overlooked, or some other kind of legal breach occurs. When an open community of different individuals is gathered, different opinions and beliefs can conflict, and they might disagree at specific points of a project. If no

consensus can be found, a fork might be the right solution, and the community splits into two different projects.

3.3 *Engagement with the Open Source Ecosystems*

In general, there are three different ways to distinguish how a company can engage with open source software: consume, contribute, and own their own open source software. There are different implementations of each of the engagements. A company might consider why to interact, how to act accordingly, and define what it means to engage successfully. All three ways have different social, technical, and legal implementations. In this chapter, those aspects will be reviewed as well as how to measure the maturity of the software and how to avoid risks.

3.3.1 Consuming Open Source

Open Source software is an inherent part of the software industry. There is a vast amount of open and free solutions available to be consumed. As described in a previous chapter, open source represents an important part of infrastructure software, desktop systems, and niche software applications. Consequently, large parts of the Internet Backbone are built with open-source software. Besides projects for general public interest, sizeable proprietary software companies such as IBM, Ford, Wal-Mart, Exxon, GM, Amazon Inc., and SAP are just a small sample of the immense scale where open software is used successfully. Within those examples, the most frequently used open operating systems are Linux and FreeBSD. Regarding infrastructure Apache Web server, MySQL and PostgreSQL are to mention here and can be counted as the most prominent ones. Whereas for desktop applications Open Office and BIND are vastly distributed (PaOS04).

Nevertheless, when consuming open source software, the applicable company needs to consider a variety of different factors such as overall costs, transition costs, and how to measure success. Those considerations need to be made before large applications are integrated and can lead to severe problems for the company from a long-term perspective.

3.3.2 Contributing to Open Source

Besides consuming open source software, proprietary software companies can also actively contribute to an open source software project. That has many advantages and can showcase the technological advantages and knowledge of a company, which is a positive message for finance and technology markets.

Another aspect of open contribution is that often, those projects work on important future enabling technologies. Consequently, if a company can have an impact on how the resulting end-product will be defined, that will lead to competitive advantages.

The proprietary software company SAP saw in 2004 the strategic and significant importance of the Eclipse SDK. The company, therefore, became one of the founding members soon and contributed to several other Eclipse projects such as jGit, eGit, Mat, and Tycho (LiSa19). There will be a more detailed overview of OSS projects where SAP was involved in the upcoming chapter.

In detail, this means that if a company implements an open-source software application, it might fit, to a certain extent, out of the box. However, it is likely that adjustments need to be made to meet the requirements. If a company can be part of the development process of a possible new standard, they can be sure to have essential features built-in from the start.

Moreover, by openly communicating which are wanted features, or even contributing code to the development community, feedback can be generated. In general, feedback from various parties leads to a higher quality of the product. Often, proactive engagement is beneficial and results in less overall development costs for the applicable company. Important to note is that acquiring the skills to implement and develop with newly released software can be decreased, for example, in the case of Kubernetes or Cloud Foundry and multi-cloud applications.

Large scale projects are often governed by independent foundations, which has several advantages over single-owned open source projects. Those foundations help govern projects within a neutral area, independently from the contributing organizations and unaffected by a possible dispute between two contributing parties.

Those initiatives are also founded to guarantee a neutral and long sustainable success of a project over an extended period.

SAP is a member of several open software sources and is actively contributing to those projects. A detailed overview of those engagements within the open-source ecosystem is part of chapter 4.

3.3.3 Owning Open Source Software

The third category with open source software engagement would be to own it. Dep-side to create the software, it could be bought from or donated by another organization. Regardless of how a company becomes the leading entity behind open-source software, there are some essential tasks to keep the project vital. Such tasks include keeping the audience engaged and motivated, establishing rules and agreements as well as keeping a well-documented and well-built system in place, in order to hold on to a good reputation.

When building community-based software, their attention should be drawn to its community. Many FOSS projects are created with free contributions from an extended group of developers. One of the advantages of open source might be that arguably, it provides a source of free labor, but that is not for granted. Those developers taking part in open communities have their motivation; they need to have a shared sense of a common goal to keep the audience successfully involved. In general, the benefits should be even, and the community should benefit as much from the project as the owner itself. Therefore, their contributions must be evaluated accordingly, and open source projects should remain open.

In order to keep an open project thriving, it must be observed that a public engagement of the project is equally important. That presence can attract new developers and can be accomplished by hosting public events, presenting panel talks, interviews, generating blog articles, or publishing to newspapers as well as publishing videos explaining basic concepts of the project. Ultimately, there are many ways to interact with the communities inside and outside a project. Importance remains on the communication of the core values and the possible significance of the project.

Besides the community aspect, another critical factor is the establishing of a project approval process: open source project communities can have several thousand participants, with contributions across different time zones. It is increasingly essential to the size of the community, to make use of an automated process wherever it is possible. Just a few processes, such as approvals, CLAs agreements, and status updates, can take up a significant amount of time if not addressed in a self-organized way.

Besides automation, there should be a library of guidelines and frequently asked questions established. Those guidelines should be easy to access and give a broad range of essential information. Within FAQs, there should be instructions on how to respond accordingly to specific issues or bugs, how to deal with difficult circumstances, and how to avoid common problems. That can help in many cases to get questions answered before a community member needs to answer individually. Not only that, but those guidelines can also help in case of a possible disagreement to find a consensus, and to set a variety of community rules to keep the environment professional and to keep the focus on the main project goals. Because such guidelines are standard within open projects, an organization can make use of already existing contribution model templates rather than having to recreate those mechanisms. The adoption of such templates can help accelerate the start of a project and might support in remaining focused on core tasks while minimizing some risks.

4 Open Source Software at SAP

SAP SE, the German ERP software provider, has been best known for its proprietary products. Nonetheless, the company has a long history of open source; this chapter will give an overview of different open source products, both used and produced. Furthermore, this chapter covers a selection of projects the company is actively contributing to.

One of the first official engagements with open source can be traced back to 1998 when SAP started to port their leading product R/3 to Linux. This engagement enabled businesses to choose their preferred operating system; with this, the company gained access to a new market segment.

Three years later, in 2001, SAP formed a set of definitions and guidelines on how to process and consume open-source software. The formalized documents contained aspects such as open-source licenses, security, and control restrictions on exports.

In 2004, the company became a founding member of the Eclipse Foundation and contributed actively to several projects from the foundation, such as jGit, eGit, Mat, and Tycho (LiSa19).

The company decided in 2008 to enable more employees to contribute to external open source projects. Therefore, a guide was established on how SAP developers should engage with external communities to stay compliant and to prevent legal problems.

Two years later, in 2010, SAP formalized further how to develop and engage with the open-source ecosystem and introduced a systematic scanning process. It became part of every project and helped the company stay compliant and obtain security goals.

By 2014, SAP open-sourced a Contributor Licenses Agreement system on GitHub, which automated the license agreement from developers to contribute to an open project. There will be further details on the open-source tool in the following section.

In the same year, SAP joined the Cloud Foundry, which marks another cornerstone in the history of the company investing in the open-source community (ClSa17).

Cloud Foundry is an open-source service platform and makes multi-cloud hosting possible; besides that, it provides different applications; for example, a tool for software development lifecycles, called BOSH.

The partnership grew from a single developer to make some pull requests, to several teams containing more than 80 employees working full time on different projects, for example the afore-mentioned BOSH application, multi-cloud hosting, API connection, and several more (SaCo18). One of the latest projects is in partnership with IBM and Asus to integrate server-less applications and Kubernetes and to enable a hybrid cloud solution.

In 2016, SAP opened a BOSH OpenStack CPI Dojo in Waldorf, Germany, in cooperation with further developers from SUSE Linux. The Dojo concept was founded at the Cloud Foundry after they observed how long it took developers to gain submitter status in an OSS project. The submitter status is an indicator that a developer is full-time, contributing to one of the Cloud Foundry projects. The Dojo provides a 6 to 12 weeks training program for a developer to acquire the necessary skills in a fast-paced environment. SAP has established worldwide 8 Dojos, where they continuously train developers and contribute to open source projects (ClDo18).

In 2017 SAP joined the Hyperledger program as a premier member. Hyperledger is a part of the Linux Foundation and develops and conducts research on blockchain technologies (HyLi17).

In the same year, SAP joined the Cloud Native Computing Foundation and the Open API Initiative (SaLi17). There will be a more detailed perspective on those engagements in the upcoming section.

In 2018, the German ERP provider entered an agreement with Google and Intel to deliver a serverless cloud solution on the public GCP cloud with significant performance improvements for customers (PaGo18).

In the same year, SAP established a centralized internal organization to regulate and oversee all open source activity. The so-called Open Source Program Office, in short OSPO, consists of a virtual and global team and has a focus on streamlining open source engagements and on being a single point of contact for OSS-related inquiries.

Moreover, the department is overseen by the CTO of SAP, which results in better integration with the company (LiSa19).

Based on those engagements, it can be observed that there is a general endeavor to establish and strengthen the contact with external open projects. In the upcoming section, there will be a more detailed review of open source projects which were launched by SAP.

4.1 *Software Tools*

This chapter will present an overview of open-source software projects initiated by the German software provider SAP. There is a wide variety of projects in this category since SAP is actively motivating and enabling its employees to contribute to open projects. Nonetheless, this section of the chapter will be focused on a selection of the most relevant open source software projects maintained by SAP.

4.1.1 Contributor License Agreement Tool

There are many ways how a company can handle legal rights for software contributions. If an internal employee is working and contributing to an internal project, the rights are most certainly owned by the employing company. However, once an external developer is contributing to a project, the question of who can claim ownership of the creation arises, primarily if the external contributor is employed at an organization or company. Even if the contribution is for an open project, intellectual ownership of the creation is still relevant.

In order for an SAP to manage software contributions to their open projects, they created an open-source tool to automate agreements: The Contributor License Agreement Tool also referred to as CLA. The open program supports the agreement process of ownership-related topics: CLA gives the agreeing parties the security that legal claims upon ownership of submitted creations are defined. The parties agree that they are willing to give up any further right and that they are legally able to contribute. It is especially essential if, for example, competing companies or their employees, in some cases, even unintended, are contributing to projects.

SAP's open CLA tool was released for the first time to the public in 2014 under an Apache License, Version 2.0. It was created in collaboration with the developer team

form GitHub. The project itself is hosted on GitHub and is openly accessible to interested individuals. It is used by a respected, large number of other companies and organizations, which forked the project and created their own version (SaCI19).

In order to have a consistent and robust agreement system for all projects, every single contribution needs to be signed off with the CLA assistant. The agreement is signed with every pull request from the project; every contributing developer needs to agree to the CLA. The software authenticates the signee and updates the status of the pull request. This process is integrated into the hosting platform; in this case, the GitHub platform (GiCI19). Therefore, an uninterrupted chain of agreements is created for the developer and the company to stay as compliant as possible.

4.1.2 SAP Cloud Platform

SAP Cloud Platform provides a single space for all SAP cloud applications. It functions as an open Platform-as-a-Service, which can be run on the SAP HANA database. The program was and is developed together with SUSE with open source technologies. Initially, it was developed within the SAP HANA Cloud Platform with the name SAP NetWeaver Cloud. Since then, the platform grew steadily, and many features and programs where added over time.

One function of the platform is to establish a connection between on-premises and cloud-hosted applications to work together, as well as third-party applications. The platform embraces open standards, including Java, JS, Node.js, and it works together with Cloud Foundry. The latter enables the platform to host multiple third-party cloud services such as AWS, MS Azure, GCP, Alibaba, to name a few (CloS17).

Due to the close relation to the Cloud Foundry and the BOSH development cycle, it supports customers in iterating on new ideas with an increased pace compared to conventional methods, and it seems the development cycle affects the go-to-market time positively. SAP provides several SDKs for different developer platforms, such as for Apple iOS and Android. Those SDKs come with documentation and additional material, and supplies developers with already existing libraries to accelerate their creation of applications (CPWi19).

The cloud platform is continuously growing and implementing new emerging technologies like Dockers and Kubernetes. Moreover, it gives customers the ability to try out different new technologies such as Blockchain, Machine Learning, IoT, Big Data, and different analytic tools.

The following applications are hosted and made available on the SAP Cloud Platform.

4.1.3 Openui5 and SAPUI5

In 2013, SAP decided to open their framework and follow a dual-license strategy. The company released openUI5 under the Apache 2.0 license. One year later, in 2014, the OpenUI5 team was open for contributions made via GitHub.

SAP UI5 is a JavaScript Web toolkit for user interfaces. The program represents a library for user interfaces and works together with SAP Fiori, which is the companies design language. SAP Ui5 is available as an open and closed source variation. Both versions are almost identical and differentiate mainly due to their different licenses. The UI5 library contains more than 200 frontend controls and follows a Model-View-Controller approach.

The program was initially developed under a proprietary license to provide a consistent design language for web applications for the company and its customers. Previously, SAP introduced new user interface libraries with each new product release. It provided many advantages to separate products and designs to keep a consistent experience across several applications and to represent a unified brand.

Various applications follow those design principles or have similar expressions enabling a consistent appearance; thus, user experience can be achieved throughout different programs. The program is built with JavaScript and XML code generated compliant HTML, which enables the program to keep its consistency across different platforms and runtimes. Further worth mentioning is its compatibility to work together with OData and JSON API’s (FiSa17).

Initially, the internal project name of the software library was “Phoenix.” In 2009 SAP developers worked on a strategy to keep a consistent UI experience over a long period of time and across different applications and product releases. Moreover, they tried to solve a problem that occurred due to the fact that the previous UI frameworks were

tightly coupled with the backend of SAP's technology stack, which became a bottleneck when briefly changing the appearance of the program. The developer team behind the software believed that the software needed to be opened to a broader and external community to increase its potential. They saw the benefit of collaborating with different software communities and sharing their creation with other projects.

Around the year 2013, as smartphones and mobile devices became increasingly crucial for businesses and private persons, the perception was shifting towards a mobile-first and mobile-friendly workplace. As a result, a variety of mobile apps where created, this trend was observable in a wide range of businesses and industries. Many companies soon had a variety of programs and different mobile applications. However, most companies tried to have the right consistency within their application appearances. Since SAP UI5 was proprietary software, customers were not permitted to implement its user interface experience in other third-party applications, which made it difficult for SAP's customers to achieve a consistent appearance. Another problem software developer encountered with the original proprietary character of the software was that feedback from partners, and even internal employees were not permitted to be implemented to improve the software because of license issues.

Another item in the list of arguments for SAP to open source was that their Ui5 application caused a shortage of talent hiring possibilities for UI departments. Since new hired developers were not familiar with the application, training costs, and getting up to speed times were higher compared to other areas. Furthermore, SAP considered the positive impact from external communities to develop the library further and expand its potential behind its primary purpose. Since the way how programs were developed shifted to a more open approach and many developers preferred to work with open frameworks in order not to be tied to one platform, SAP needed to change their approach to stay relevant in the developer communities, especially outside of the SAP ecosystem.

SAP Ui5 remained as a SAP internal build version of the same software. It is a downstream product of the OpenUi5 and marginally enhanced with some additional features. SAP Ui5 is made available to SAP customers bundled with other software components and comes together with a service agreement. There is a full team within

SAP working on Ui5 to improve its functionalities. On the other hand, OpenUi5 is hosted on GitHub; it has several versions, nightly builds, betas, and final releases. It has an active community with over 40,000 Comments (Dec. 2018), 360 releases, and 56 Branches. The GitHub provided bug-tracking program is additionally used to improve the software.

For SAP, this project represents a success in various ways. Overall it enables the company, together with a community around the product, to contribute and further develop the program.

4.1.4 Chevrotain

Chevrotain is similar to OpenUi5 and represents a code parsing library for JavaScript and NodeJS. It was developed by SAP and is licensed under an Apache 2.0 license and released to the public in 2015 via GitHub. Despite the large variety of parser building toolkits available, the development was initiated by one SAP employee who wanted to develop his own version with specific customizations. Since the public release, Shahar Soel, also known under his developer name bd82 only commits to the project in his own time and, therefore, strictly separates the work he is doing at SAP and his commitment within the open program. The community around that library is significantly smaller compared to the other projects. In December of 2018, the project counted on GitHub 1,568 submitted commits, 22 branches, 122 releases, and 20 active contributors (OpSa18).

4.1.5 SNAP!

Snap! is a visual programming language based on connected block segments. It was developed by SAP in cooperation with the University of Berkeley in 2011. The language aims to support the learning and teaching of basic programming concepts for any age. It was inspired by Scratch, which was developed at the MIT Media Lab (ScWi18). The software is distributed under an open-source AGPL v1 license and was initially developed by Brian Harvey and Jens Moenig in 2011 (SnWi18).

The software is written in JavaScript, and the interface is accessible via any standard web browser. Unlike Scratch, Snap! has enriched features like more advanced concepts, allowing to teach more complex content to even advanced software

students. (SnWi18). The programming language is helping to bridge the gap between several educational target groups. Besides Berkeley and SAP, many other educational and private institutions are using the software to teach and to discuss different architectural concepts and basic programming fundamentals.

4.1.6 BUILD

Build is a SAP internally developed design tool, which also allows a non-technical user to create interactive prototypes. It utilizes different input types and can automatically convert paper doodles into a functioning mockup prototype. The program was launched in 2014 under an Apache 2.0v license. It is made available and hosted on GitHub. Besides the source code, a functioning version can be accessed with any standard web browser (BuGi14).

Furthermore, the program has the capability to use real sample data and to tap into a broad technology stack such as Mongo.DB, Express, AngularJS, and Node.js. It follows the SAP Fiori design guidelines and can access UI frameworks such as OpenUI5 (BuBu18).

4.2 *Contributions*

SAP is part of several open source and general software foundations. Each of them serves a specific purpose and supports the company in various ways. Working in a community on potentially disruptive new technologies seems to appeal to the software company. A contribution has several dynamics; however, two aspects are to be considered: The more companies are working on innovation, the more likely it is that they get established. The other aspect is to predict specific trends. Companies with extensive research and development departments tend to be more accurate about such predictions.

In general, contributing companies are not just donating funds to a foundation, but instead, they are actively engaged in the development of innovation, contributing core competencies in various fields of technology. Another advantage of an external foundation is the neutrality and agility they can operate. Some foundations need to exist over several decades to deliver tangible results.

Many of the following foundations seem to work on technologies that might enable the next generation of future applications and technologies. To be part of such foundations comes with a high price. Therefore, companies are making sure that the investment is placed carefully in order to extract potential wins in the form of skills, knowledge, and future-ready applications.

4.2.1 Linux Foundation

The Linux Foundation is a non-profit trade organization, which has its origins in 1993. In the same year, the first Linux mailing list was introduced and received global attention. In 2000 the foundation was officially established after the merge of two large Linux groups.

The foundation serves various purposes: it helps to manage Linux communities, oversees projects, fosters innovation, and forms a neutral place for Linux and open-source projects to develop and evolve. The website Linux.com is hosted by the organization as well, which is developing into a platform for tutorials and guidelines on how to engage with the open source ecosystem.

For the last couple of years, the foundation has been broadening its scope of interest to new emerging open source technologies and is increasingly becoming an umbrella organization for a set of different foundations. SAP is engaging with a variety of those newer foundations, which will be part of the upcoming sections. Those new areas include technologies such as blockchain, cloud computing, Kubernetes, Dockers, drones, and a range of further projects like OpenAPI and Open Container Initiative.

SAP is a member of the foundation since 2017; the membership presents for the company an essential step for its open-source strategy. There are several benefits, and it is seen as a reliable indicator of the increasing importance of open source technologies in the industry.

The **Node.js Foundation** is part of the Linux Foundation and was established in 2015. The organization was founded to support the adoption of the Node.js and other modules, which is accomplished by an open governance model and therefore encourages engagement and participation from partners and members. The open-governed aspect should also guarantee the long-term success of the organization and possibly enables the continued improvement over an extended period.

Node.js itself is a JavaScript run-time environment that operates cross-platform and runs JavaScript outside of an internet browser. It has an MIT open-source license and supports a variety of operating systems.

The foundation is essential, aside from its cross-platform capabilities, with their applied skills and development in JavaScript technologies. JavaScript is a core technology in the SAP Cloud Platform, and it is essential for the company to have that technology up to date, and upcoming changes implement as early as possible to give their customers the best experience.

Google donated the **Cloud Native Computing Foundation** to the Linux Foundation in 2015. It is an open-source and not-for-profit organization to build a collaborative community to improve and develop cloud and cloud-related technologies primarily in the area of application speed and scaling. The foundation established a variety of projects to serve those purposes, and a large community has formed around the organization to distribute and to learn the new evolutions in the fast-growing area of

cloud computing. Since 2017, SAP is a member of the foundation and is jointly working on its improvements alongside companies such as Amazon, MS, Oracle, VMware, and many more. The decision of SAP to join the foundation is supporting the company in several ways, but one crucial area is the development of a hybrid cloud model, which SAP is already implementing, it can still be further enhanced to give customers an extra reliable and faster integration from on-premises and cloud-hosted applications (SaCN17).

The **CNCF** is working on a broad set of essential projects which are of potential value to SAP. The projects contain **Kubernetes**, **Container,** or **Envoy**. These projects are concentrated around the possibility of exploring the capabilities of cloud computing and are seen as critical areas and seed technologies for further improvements in that particular area. Kubernetes, for instance, is one of the sub-projects of the foundation, it is an open-source framework for the automation of deployment and control of applications using containerization and clustering. The orchestrating tool is also used as an alternative to virtualized runtimes. The software makes it possible to run web applications in different environments, such as on-premises and cloud (KuWi18).

The **Hyperledger** project is founded and governed by the Linux Foundation and was established to improve blockchain technologies and provide guidelines and educational courses for interested organizations. The project is concentrated on open source blockchain technologies and combines a variety of tools and communities to enable synergy effects, and it helps the technology mature and grow. SAP is part of the project in addition to several other industry-leading cooperation such as IBM, Cisco, Fujitsu, Intel, Red Hat, and VMware (HyWi18).

The **OpenAPI Initiative** is part of the Linux Foundation and was a donation of SmartBear Software. The original project name was Swagger Specification and consisted mainly of a set of rules and guidelines to formalize the APIs. The OpenAPI initiative helps to uniform the way REST APIs are defined. (AbOp16). REST APIs is short for Representational State Transfer; they provide guidelines for establishing web services and connections between applications (ReWi18).

SAP joined the initiative in 2017 and thus further strengthened their effort on improving cloud technologies. For SAP, the initiative represents a cornerstone; it supports the

company's development of modern applications and development environments. In that sense, cloud-native applications are increasingly built upon microservices, which rely heavily on APIs.

The OpenAPI Initiative is working on ways to improve data exchange between all kinds of different programs and devices, which is especially essential when it comes to IoT technologies. Those devices rely on seamless communications between various programs, third-party applications, and other devices to work efficiently. SAP supports that course and has opened their web-based applications such as S/4HANA, S/4HANA Cloud, and further SaaS web products to support the REST full standards (ApSp18). Customers can manage their APIs with the SAP API Business Hub and the API Manager, which gives them a centralized place to receive real-time data about API usage (ApCl18).

4.2.2 Apache Software Foundation

The Apache Software Foundation is an open, not-for-profit trade decentralized organization to support open-source software; it was established in 1999 (ApWi18). The foundation governs open source licenses and gives training, guidance, and support in the establishing process of open-source software. As for the organizational structure of the foundation, it is an independent entity and organized as a meritocracy. Companies can take part via memberships that are only handed to volunteering organizations that support Apache projects actively. Their focus is on the continuous development in areas such as web servers, Big data, and, therefore, applications like Hadoop, KAFKA, and spark, which are becoming increasingly important.

4.2.3 Eclipse Foundation

The Eclipse Foundation is a not-for-profit, independent organization to support open-source software, frameworks and development environments. It holds more than 275 member organizations and has established a vendor-neutral, transparent, open, and global community. It supported more than 350 open source projects in 2018 in a broad range of technologies such as business intelligence, cloud computing, IoT, and web tool automation development. The initial project of the foundation was the Eclipse SDK platform. The software is a development environment supporting a significant number of different developer languages (EcWi18). SAP is one of the founding members and

has been on board since 2004. Together the organizations are researching in a broad range of projects, for example, jGit and EGit. In an upcoming section of this thesis, there will be a review of those Git software projects produced by SAP and partners.

4.2.4 OpenStack Foundation

OpenStack is an open platform for cloud computing technologies; it provides an Infrastructure-as-a-service (IaaS). It primarily focuses on virtual runtime software support. The project was initially founded jointly by NASA and Rackspace in 2010. Two years later, in 2012, the project was opened to the public and accessible to business partners. Since then, the project serves more than 60,000 individual members globally in more than 180 countries. In 2016 OpenStack established a new organizational structure and had been governed by the OpenStack Foundation ever since. The foundation is vital for SAP in business application relevant areas such as networking, storage management, and identification (OpWi18).

4.2.5 Cloud Foundry Foundation

The Cloud Foundry Foundation is an independent and not-for-profit organization that governs a range of open source cloud-related software applications.

Initially, the Cloud Foundry Foundation was developed by Chris Richardson in 2008, which was acquired by SpringSource the following year in 2009 (CFWi18). However, SpringSource was sold shortly after the acquisition of VMware. Nonetheless, the original open-source software application, Cloud Foundry, was further developed and produced despite those events. The software allows to host applications on multiple cloud providers and was officially released to the public in 2011 (ClWI18). VMware initiated a spin-off together with General Electric to establish the company Pivotal to monetize the product (WiVM18). In 2015 VMware decided to release the Cloud Foundry project as a collaborative project and granted all rights to the Linux Foundation. Due to the neutral character of the foundation, the project was able to develop further with no restrictions based on company rules or be converted into closed source software for profit.

The foundation gathered vast interest around software companies, and SAP became a member in 2017 and opened a Cloud Foundry Dojo in Walldorf, Germany, where several employees from both organizations work jointly on cloud software (ClSA18).

4.2.6 jGit

The Git application supports the management and tracking of changes submitted to a software project. It is an essential tool for open source communities and was initially developed by Linus Torvalds in 2005. Different versions of Git became part of development tools such as the Eclipse IDE (GiWi218).

jGit supports working in a Java environment with communications between Eclipse and a Git SCM. It can pull and push requests from code repositories and work with several branches of the project and commit them back to the project. Therefore, it handles a part of the organizational work around a software project. jGit itself is composed in Java and makes it suitable for developers to update their Git repositories without the need to use another external compiler library (EcjG17).

In 2008, Shawn Pearce and four other SAP members joined the Eclipse eGit/jGit project alongside engineers from other companies. Two years later, in 2010, Eclipse started to use jGit within their IDE.

4.2.7 Cloud Foundry

Cloud Foundry is a platform as a Service (PaaS) and is openly governed by the Cloud Foundry Foundation and its partner members. The platform provides a multi-cloud application, which allows running a wide variety of software languages, browser-based with the integration of several instances and programs (CfWi18).

In 2018 SAP had 68 developers working together with Cloud Foundry on both the core and extensions to develop new projects, contribute actively to existing software, and enabling the further integration of open source software around SAP. Most of the developers contribute full time to those projects, and besides contributing code also pull requests are reviewed (SaCl18).

4.2.8 Kubernetes

Kubernetes help to reduce the complexity with containerized frameworks and works alongside the SAP Cloud Platform to manage several cloud applications. It is an Open Source project which was initially founded by Google Inc. in 2014 (KubW18). It was created to manage a large scale of applications across different servers, to handle a scalable amount of data without compromising a running system. In 2015, Google teamed up with the Linux Fountain to create an independent organization: The Cloud Native Computing Foundation - CNCF. The foundation oversees the open API initiative and helps to formalize the project development and to make contributions from external individuals and companies possible (KuTe16). SAP became a member of the CNCF in this early stage and therefore strengthened itself to establish a Cloud-as-a-Service model besides its Platform-as-a-Service offers.

Since then, to project has reached considerable interest across the cloud industry, and in 201, it was one of the top 10 projects on GitHub and one of the highest contributors to the Linux Foundation (KuCo17).

5 Conclusion

To conclude this thesis, few reasons can be isolated to answer the initial question of why a proprietary software company engages with open-source software. It needs to be acknowledged that not all contributing factors of a company deciding whether to engage with the open-source software ecosystem or not, can be considered. Instead, due to the observations on the matter, some different aspects of why an engagement might be beneficial are pointed out here. In this thesis, the focus was set on the proprietary software provider SAP SE to generate a profound understanding of possible motivating factors of a single case. Therefrom follows concluding aspects that contributed to their structural shift to increase open source activities.

One of the initial engagements with open source software was the porting of the SAP R/3 product to Linux in 1998. Due to the early success of providing software to the open-source market, the company started to explore other areas where Linux could be adopted and might lead to a sales increase.

Open and different forms of co-innovation represent the most compelling argument for engaging with open source for Peter Giese, director of SAP's Open Source Program Office (SaLi19). This form of collaboration can lead to a set of positive benefits for a company.

Prahalad and Hamel pointed out (1990) that the concentration of a company on its core competencies increases its chances of succeeding in markets (CCWi90). Furthermore, it increases innovation capabilities in certain niche technology areas. The collaborative combination of knowledge and core competencies can help a group of companies with research and development projects. Due to the collaborative effort of companies and open communities, the chance to set a new standard increase with the size and number of the participating members. If a critical adoption of new technologies and formats is reached, it can establish new standards comparable to the precipices introduced by Utterback and Abernathy of the Dominant Design (DDWi19).

Another aspect that is suiting to proprietary software providers is to be close to so-called innovation hubs. In taking part in an open developer community that is working

on the development of new standards such as OpenAPI, a company can influence and adapt important technology as early as possible.

Besides the increase of innovation capabilities and eliminating the risk of unforeseen disruptive technologies, another decisive factor is the access to talent pools. Companies are in constant competition for new potential hires in key technologies. It is seen as one of the mission-critical tasks of a company to attract and hire the best talents in an industry in order to stay relevant to the market. Open source communities have proven themselves as an excellent reference for human resources. Besides having a collective understanding of a specific topic, members of open communities are more likely to be higher intrinsically motivated. That can have a two-sided effect: besides having access to a talented community, it also reduces the costs of acquiring new employees.

Gray and Balmer pointed out in their studies the importance of a managed corporate image and reputation to be observed by customers as an innovative organization. By actively engaging with open source communities, a positive reputation can be maintained and increased as a technological advocate company. However, when consuming open-source software, companies stated that moral obligations to invest back to the open-source ecosystem is part of using open source (OSTJ03).

Since 1998, when SAP made its products for Linux available, the company has engaged with the open-source ecosystem. However, the company is instead recognized as a traditional proprietary software company. That is likely due to two reasons: first, the majority of products of the ERP company are only able to be obtained as closed source software. The company can be regarded as successful with this strategy in an economic aspect. The second reason for the lack of awareness for SAP's open-source engagements is that insufficient marketing campaigns were released to promote their free products. This is also due to the different characteristics of how open source products are distributed in comparison to proprietary products.

Nonetheless, SAP decided to advert its engagement more and to streamline its engagements. Because of this, the company introduced a new department, the open-source program office (OSPO). The department consists of globally located members from different entities and is a centralized place for all topics surrounding open-source

software. It seems to be of increased importance for technologies like cloud computing and artificial intelligence, which are predominantly developed by open communities. It is likely that this trend contributed to the shift of the industry towards open source technologies.

Finally, one can conclude that former strictly proprietary software providers are increasingly engaging with open source technologies to increase their competitive advantage (CAWi87). This is also due to the increased importance of open technologies as described above and to the ubiquitous integrated character of those emerging technologies, which will eventually bypass the current trend based on the technology life cycle (LifC09).

It is to point out that an engagement with the open-source ecosystem is often observable on a broad-spectrum, including consuming, owning, and contributing to open projects. Although there is already a long history of such engagements, they have, so far, not been promoted publicly in the same manner as we can observe it in the recent decade of 2010.

6 Glossary

AI	Artificial Intelligence
ASF	Apache Software Foundation
AWS	Amazon Web Service
B2B	Business to Business
B2C	Business to Consumer
BASIC	Beginner's All-purpose Symbolic Instruction Code
COBOL	Common Business-Oriented Language Customer
CRM	Relationship Management
ERP	Enterprise Resource Planning
FOSS	Free and Open Source Software
FSM	Free Software Movement
GCP	Google Cloud Platform
GNU	GNU is not UNIX
HCP	High-Performance Computing
HRM	Human Resource Management
IaaS	Infrastructure-as-a-Service
IDE	Integrated Development Environment
LOB	Line of Business
ML	Machine Learning
OSI	Open Source Initiative
PaaS	Platform-as-a-Service
SAP	Plasmid-Based
SCM	Supply Chain Management
SME	Small and mid-sized companies
SaaS	Software-as-a-Service

7 References

AbOp16. (2016). *OpenAPIs - About*. Retrieved 12 09, 2018, from https://www.openapis.org/about

AbOp18. (2018, 2019 05). *Abas-erp.com - closed-source-open-source-erp*. Retrieved from https://abas-erp.com/en/news/closed-source-open-source-erp: 28

AbSa18. (2017). *blogs.sap.com - Overview of ABAP in SAP Cloud Platform*. Retrieved 11 26, 2018, from https://blogs.sap.com/2017/09/26/overview-of-abap-in-sap-cloud-platform/

AbWi18. (2018). *en.wikipedia.org - ABAP*. Retrieved 11 25, 2018, from https://en.wikipedia.org/w/index.php?title=ABAP&oldid=867639666

AdER15. (2015). *ERP SoftwareBlog - The Advantages and Disadvantages of Cloud and On-Premise ERP Systems*. Retrieved 04 22, 2019, from https://www.erpsoftwareblog.com/2015/12/the-advantages-and-disadvantages-of-cloud-and-on-premise-erp-systems/

AfWi18. (2018). *Wikipedia - Affero*. Retrieved 12 11, 2018, from https://en.wikipedia.org/w/index.php?title=Affero_General_Public_License&oldid=845529400

AiGo18. (2015). *ai.googleblog.com - TensorFlow - Google's latest machine learning system, open sourced for everyone*. Retrieved 12 30, 2018, from https://ai.googleblog.com/2015/11/

AmWi16. (2016). *Wired.com - Amazon's Giving away the AI Behind its Product Recommendations*. Retrieved 12 16, 2018, from https://www.wired.com/2016/05/amazons-giving-away-ai-behind-product-recommendations/

AnWi18. (2018). *Android - Wikipedia*. Retrieved 11 15, 2018, from https://en.wikipedia.org/w/index.php?title=Android_(operating_system)&oldid=868047026

ApCl18. (2018). *Cloudplatform.sap.com - SAP Cloud Platform API Management*. Retrieved 05 12, 2019, from https://cloudplatform.sap.com/capabilities/product-info.SAP-Cloud-Platform-API-Management.e39ffff0-6b34-4611-9989-20da901caa47.html

ApcW18. (2018, 08 12). *Wikipedia - Apache License*. Retrieved 12 10, 2018, from https://en.wikipedia.org/w/index.php?title=Apache_License&oldid=873376586

ApRi07. (2007). *Richart Stallman - The Problems with older versions of the Apple Public Source License (APSL)* . Retrieved 01 16, 2019, from https://www.gnu.org/philosophy/historical-apsl.en.html

ApSp18. (2018). *SAP - API*. Retrieved 05 12, 2019, from https://api.sap.com

ApTo17. (2017). *Appsruntheworld.com - ERP Software*. Retrieved 04 24, 2019, from https://www.appsruntheworld.com/top-10-erp-software-vendors-and-market-forecast/

ApWi18. (2018). *Wikipedia - Apache Software Foundation*. Retrieved 12 09, 2018, from https://en.wikipedia.org/w/index.php?title=The_Apache_Software_Foundation&oldid=871213657

AtBu16. (2016). *businessinsider - This SAP president has a fabulous career because he spent one sad and lonely holiday at work*. Retrieved 11 23, 2018, from https://www.businessinsider.de/from-lonley-holiday-to-sap-president-2016-3?r=US&IR=T

BSDW18. (2018). *Wikipedia -BSD Licenses*. Retrieved 12 13, 2018, from https://en.wikipedia.org/w/index.php?title=BSD_licenses&oldid=873127188

BuBu18. (2018, 10 12). *Build.me - Introduction*. Retrieved 12 16, 2018, from https://www.build.me/blog

BuGi14. (2014). *Github - BUILD-Overview*. Retrieved 12 14, 2018, from https://github.com/SAP/BUILD/wiki/BUILD-Overview

CAWi87. (2019). *Wikipedia.org - Competitive Advantage*. Retrieved 05 29, 2019, from https://en.wikipedia.org/w/index.php?title=Competitive_advantage&oldid=877552305

CCWi90. (2019). *Wikipedia.org - Core Competency*. Retrieved 05 26, 2019, from https://en.wikipedia.org/w/index.php?title=Core_competency&oldid=891869621

CFWi18. (2018). *Wikipedia - Cloud Foundry*. Retrieved 12 09, 2018, from https://en.wikipedia.org/w/index.php?title=Cloud_Foundry&oldid=866855488

CfWi18. (2018). *Wikipedia.org - Cloud Foundry*. Retrieved 12 13, 2018, from https://en.wikipedia.org/w/index.php?title=Cloud_Foundry&oldid=863673611

ClDo18. (2018, 10 12). *Cloud Foundry - Dojo Engieering*. Retrieved 05 05, 2019, from https://www.cloudfoundry.org/engineering/

CloS17. (2017, 12 1). *SAP - Cloud Platform* . Retrieved 05 10, 2019, from https://cloudplatform.sap.com/index.html

CloW18. (2018). *Cloud Computing - Wikipedia*. Retrieved 11 20, 2018, from https://en.wikipedia.org/w/index.php?title=Cloud_computing&oldid=869660970

ClSa17. (2017, 10 2). *Cloudplatform.sap.com - Cloud Foundry*. Retrieved 11 23, 2018, from https://cloudplatform.sap.com/enterprise-paas/cloudfoundry.html

ClSA18. (2018). *Cloudplatform.sap.com - SAP Cloud Platform Multi-Cloud and Cloud Foundry*. Retrieved 12 09, 2018, from https://cloudplatform.sap.com/enterprise-paas/cloudfoundry.html

ClWi18. (2018). *Cloud Foundry Wikipedia*. Retrieved 11 03, 2018, from https://en.wikipedia.org/w/index.php?title=Cloud_Foundry&oldid=866855488

ClWI18. (2018, 10 2). *Wikipedia.org - Cloud Foundry*. Retrieved 12 09, 2018, from https://en.wikipedia.org/w/index.php?title=Cloud_Foundry&oldid=866855488

CoFr18. (2018). *Wikipedia - Comparison of free and open-source software licenses*. Retrieved 12 13, 2018, from https://en.wikipedia.org/w/index.php?title=Comparison_of_free_and_open-source_software_licenses&oldid=873409120

CoWi18. (2018). *COBOL Wikipedia*. Retrieved 11 03, 2018, from https://en.wikipedia.org/w/index.php?title=COBOL&oldid=866770674

CoWi18. (2018). *Wikipedia - Copyleft*. Retrieved 12 09, 2018, from https://en.wikipedia.org/w/index.php?title=Copyleft&oldid=867678706

CPWi19. (2019). *Wikipedia - SAP_Cloud_Platform*. Retrieved 05 10, 2019, from https://en.wikipedia.org/w/index.php?title=SAP_Cloud_Platform&oldid=887997567

DDWi19. (2019). *Wikipedia.org - Dominant Design*. Retrieved 05 25, 2019, from https://en.wikipedia.org/w/index.php?title=Dominant_design&oldid=868941106

DocCr18. (2018). *www.crunchbase.com - Docker*. Retrieved 11 26, 2018, from https://www.crunchbase.com/organization/docker#section-overview

DocWi18. (2018). *Wikipedia - Cloud Foundry*. Retrieved 11 26, 2018, from https://en.wikipedia.org/wiki/Cloud_Foundry

DoWi18. (2018). *Wikipedia - Dot-com bubble*. Retrieved 11 11, 2018, from https://en.wikipedia.org/w/index.php?title=Dot-com_bubble&oldid=867445707

EcjG17. (2017). *Eclipse.org - jGit*. Retrieved 05 24, 2019, from https://www.eclipse.org/jgit/

EcWi18. (2018). *Eclipse Foundation Wikipedia*. Retrieved 11 03, 2018, from https://en.wikipedia.org/w/index.php?title=Eclipse_Foundation&oldid=866377135

EcWi18. (2018). *Wikipedia - Eclipse Foundation*. Retrieved 12 09, 2018, from https://en.wikipedia.org/w/index.php?title=Eclipse_Foundation&oldid=866377135

EnWi18. (2018, 6 2). *Wikipedia - ENIAC* . Retrieved 11 03, 2018, from https://en.wikipedia.org/w/index.php?title=ENIAC&oldid=866436282

EnWi18. (2018). *Wikipedia - Enterprise resource planning*. Retrieved 02 10, 2019, from https://en.wikipedia.org/w/index.php?title=Enterprise_resource_planning&oldid=881875356

ErRa07. (2007, 01 01). *rarenewspapers - Open Innovation*. Retrieved 07 26, 2018, from http://www.rarenewspapers.com/view/606375?imagelist=1

FiSa17. (2017, 10 2). *Sap.com - Best UI5 APP Ever*. Retrieved 05 10, 2019, from https://blogs.sap.com/2017/01/13/best-ui5-app-ever-.../

FitzT06. (2006). Fitzgerald, B. - The transformation of open source software. *MIS Quarterly, 30*(3), 587–598.

FoWi18. (2018). *Fork Software Development Wikipedia*. Retrieved 11 03, 2018, from https://en.wikipedia.org/w/index.php?title=Fork_(software_development)&oldid=863113214

FrRi07. (2015). *Richard Stallman - Why Open Source misses the point of Free Software*. Retrieved 01 17, 2019, from https://www.gnu.org/philosophy/open-source-misses-the-point.en.html

FrWa18. (2018). *Wikipedia - Freeware*. Retrieved 12 11, 2018, from https://en.wikipedia.org/w/index.php?title=Freeware&oldid=870022180

FrWi18. (2018, 08 26). *Wikipedia - Free Software Initiative*. Retrieved 09 20, 2018, from https://en.wikipedia.org/w/index.php?title=Alternative_terms_for_free_software&oldid=820618860

GiCl19. (2019). *Github.com - cla-assistant*. Retrieved 05 08, 2019, from https://github.com/cla-assistant/cla-assistant

GiWi18. (2018, 08 21). *Wikipedia - Git*. Retrieved 11 03, 2018, from https://en.wikipedia.org/w/index.php?title=Git&oldid=864178382

GiWi218. (2018, 12 1). *Wikipedia - Structure of Git Systems*. Retrieved 05 23, 2019, from https://en.wikipedia.org/w/index.php?title=Git&oldid=896664456

GnFr96. (1996). *GNU.org - The Free Software Definition*. Retrieved 12 11, 2018, from http://www.gnu.org/philosophy/free-sw.en.html

GnWi18. (2018). *Wikipedia - GNU General Public License*. Retrieved 12 11, 2018, from https://en.wikipedia.org/w/index.php?title=GNU_General_Public_License&oldid=873125522

HalH98. (1998). *Carl Shapiro, Hal Varian - Information Rules: A Strategic Guide to the Network Economy.* US: Harvard Business School Press.

HaWi18. (2018). *Wikipedia - Halloween documents*. Retrieved 01 17, 2019, from https://en.wikipedia.org/w/index.php?title=Halloween_documents&oldid=861165138

HipMI05. (2005). *Democratizing innovation. Cambridge.* Massachusetts: MIT Press - von Hippel, E.

HyLi17. (2017). *www.linuxfoundation.org - Hyperledger Welcomes SAP as Premier Member*. Retrieved 11 23, 2018, from https://www.linuxfoundation.org/press-release/2017/03/hyperledger-welcomes-sap-as-premier-member/

HyWi18. (2018). *Wikipedia - Hyperleger*. Retrieved 12 09, 2018, from https://en.wikipedia.org/w/index.php?title=Hyperledger&oldid=872076238

InfM90. (1990). *Information Technology Implementation Research: A Technological Diffusion Approach - R. B. Cooper, R. W. Zmud* (36/2 ed.). Management of Science.

Isols18. (2016). *www.iso.org - develop and publish International Standards*. Retrieved 11 26, 2018, from https://www.iso.org/standards.html

JoBa18. (2018). *SAP Blog Open Source Monday Baker, Jonathan*. Retrieved 11 03, 2018, from https://blogs.sap.com/2018/03/19/open-source-monday-is-the-license-important/

JoOp10. (2010). *Hedman, Jonas - EVOLUTION OF BUSINESS MODELS: A CASE STUDY OF SAP*. Retrieved 11 11, 2018, from http://openarchive.cbs.dk/bitstream/handle/10398/8725/Jonas_Hedman_2.pdf?sequence=1

JosT13. (2013). *Some Simple Economics of Open Source - Josh Lerner, Jean Tirole* (02 ed.). The Journal of Industrial Economics.

KubW18. (2018). *Wikipedia.org - Kubernetes*. Retrieved 10 31, 2018, from https://en.wikipedia.org/w/index.php?title=Kubernetes&oldid=866646107

KuCo17. (2017, 09 29). *Containerjournal.com - Kubernetes*. (containerjournal) Retrieved 11 03, 2018, from https://containerjournal.com/2017/09/29/sap-commits-kubernetes-container-orchestrator/

KuTe16. (2016). *Techcrunch.com - Kubernetes*. Retrieved 10 31, 2018, from https://techcrunch.com/2015/07/21/as-kubernetes-hits-1-0-google-donates-technology-to-newly-formed-cloud-native-computing-foundation-with-ibm-intel-twitter-and-others/

KuWi18. (2018). *Wikipedia - Kubernetes*. Retrieved 12 09, 2018, from https://en.wikipedia.org/w/index.php?title=Kubernetes&oldid=872771944

LifC09. (2009). *Wikipedia.org - Technology_life_cycle*. Retrieved 06 01, 2019, from https://en.wikipedia.org/w/index.php?title=Technology_life_cycle&oldid=898111673

LiGn10. (2012, 09 25). *Stallman, Richard- gnu.org - Linux and GNU*. Retrieved 10 15, 2018, from https://www.gnu.org/gnu/linux-and-gnu.en.html

LinW18. (2018). *Wikipedia.org - Linux*. Retrieved 11 15, 2018, from https://en.wikipedia.org/w/index.php?title=Linux&oldid=868885407

LiSa19. (2019). *The Linux Foundation - SAP: One of Open Source's Best Kept Secrets*. Retrieved 05 04, 2019, from https://www.linuxfoundation.org/blog/2019/01/sap-one-of-open-sources-best-kept-secrets/

LiSt18. (2018). *Linux Market Share - gs.statcounter.com*. Retrieved 11 15, 2018, from http://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-201702-201802

LiWi18. (2018). *Wikipeadia - Linus's Law*. Retrieved 04 28, 2019, from https://en.wikipedia.org/w/index.php?title=Linus%27s_Law&oldid=875114029

MoOp18. (2018). *Wikipedia - Mobile operating system*. Retrieved 04 13, 2019, from https://en.wikipedia.org/w/index.php?title=Mobile_operating_system&oldid=891609789

MPLW18. (2018). *Wikipedia - Mozilla Public License*. Retrieved 12 13, 2018, from https://en.wikipedia.org/w/index.php?title=Mozilla_Public_License&oldid=866467919

OpAi18. (2018). *Wikipedia - OpenAi*. Retrieved 12 16, 2018, from https://en.wikipedia.org/w/index.php?title=OpenAI&oldid=872793154

Open07. (2007, 03 22). *opensource.org*. Retrieved 11 03, 2018, from Open Source Initiative: https://opensource.org/docs/definition.php

OpeWi18. (2018). *en.wikipedia.org - OpenAPI*. Retrieved 11 26, 2018, from https://en.wikipedia.org/w/index.php?title=OpenAPI_Specification&oldid=867550802

OpSa18. (2018, 12 2). *Blogs.sap.com - Open Source Monday Chevrotain*. Retrieved 05 10, 2019, from https://blogs.sap.com/2018/02/04/open-source-monday-chevrotain/

OpSt99. (1999). *Stallman, Richard- Open Sources: Voices from the Open Source Revolution.* O'Reilly & Associates, Inc. Retrieved from https://www.gnu.org/gnu/thegnuproject.en.html

OpTh10. (2010). *Therigister.co.uk - Nasa Rackspace Openstack*. Retrieved 12 13, 2018, from https://www.theregister.co.uk/2010/07/19/nasa_rackspace_openstack/?page=2

OpTh16. (2016). *opensource.google.com - Thirdparty Licenses*. Retrieved 12 11, 2018, from https://opensource.google.com/docs/thirdparty/licenses/

OpWi18. (2018). *Wikipedia OpenStack*. Retrieved 12 09, 2018, from https://en.wikipedia.org/w/index.php?title=OpenStack&oldid=870677124

OrWi18. (2018). *Oracle Corporation - Wikipedia*. Retrieved 11 11, 2018, from https://en.wikipedia.org/w/index.php?title=Oracle_Corporation&oldid=868216249

OsiO18. (2018). *opensource.org - The Open Source Definition Version 1.9*. Retrieved 11 26, 2018, from https://opensource.org/osr-rationale

OSOp16. (2016). *opensource.com - Robin Muilwijk - Top 5 open source web servers*. Retrieved 01 16, 2019, from https://opensource.com/business/16/8/top-5-open-source-web-servers

OSTJ03. (2003). The Journal of Systems and Software - Open source software: an evaluation. *Alfonso Fuggetta, 66*, 70-90.

PaGo18. (2018). *cloud.google.com - Available first on Google Cloud: Intel Optane DC Persistent Memory*. Retrieved 11 23, 2018, from https://cloud.google.com/blog/topics/partners/available-first-on-google-cloud-intel-optane-dc-persistent-memory

PaOS04. (2004). Paul Kavanagh- Where Open Source Is Successful. In *Open Source Software Implementation and Management* (pp. Pages 19-40). Digital Press.

PhGn85. (85). *gnu.org - Philosophy*. Retrieved 12 5, 2018, from http://www.gnu.org/philosophy/free-sw.en.html

R3Wi18. (2018). *Wikipedia - SAP R/3*. Retrieved 02 10, 2019, from https://en.wikipedia.org/w/index.php?title=SAP_R/3&oldid=861427978

ReWi18. (2018, 12 2). *Wikipedia - Representational State Transfer*. Retrieved 12 09, 2018, from https://en.wikipedia.org/w/index.php?title=Representational_state_transfer&oldid=872203836

RiIn17. (2017). *Rightsdirect.com*. Retrieved 11 03, 2018, from https://www.rightsdirect.com/international-copyright-basics/

SaCl18. (2018). *Sap.com - SAP Cloud Platform*. Retrieved 12 18, 2018, from https://news.sap.com/2017/10/sap-cloud-platform-and-open-source-all-in/

SaCl19. (2014). *SAP - CLA-assistant*. Retrieved 05 08, 2019, from https://cla-assistant.io

SaCN17. (2017). *Cncf.io - Cloud Native Computing Foundation Welcomes SAP As Platinum Member*. Retrieved 12 09, 2018, from https://www.cncf.io/announcement/2017/10/11/cloud-native-computing-foundation-welcomes-sap-platinum-member/

SaCo18. (2018). *cloudplatform.sap.com*. Retrieved 11 23, 2018, from https://cloudplatform.sap.com/cloudfoundry/cf-contributions.html

SaHi18. (2018). *www.sap.com - SAP: A 46-year history of success*. Retrieved 11 23, 2018, from https://www.sap.com/corporate/en/company/history.2001-2010.html#2001-2010

SaLi17. (2017). *www.linuxfoundation.org - In Joining Cloud Native Computing Foundation, SAP Steps Up Its Open Source Commitment*. Retrieved 11 23, 2018, from https://www.linuxfoundation.org/blog/2017/09/joining-cloud-native-computing-foundation-sap-steps-open-source-commitment/

SaLi19. (2019). *Linuxfoundation.org - SAP: One of Open Source's Best Kept Secrets*. Retrieved 05 26, 2019, from https://www.linuxfoundation.org/blog/2019/01/sap-one-of-open-sources-best-kept-secrets/

SAWi18. (2018). *SAP ERP - Wikipedia*. Retrieved 11 11, 2018, from https://en.wikipedia.org/w/index.php?title=SAP_ERP&oldid=867817710

SaWi18. (2018). *Wikipedia - SAP SE*. Retrieved 12 15, 2018, from https://en.wikipedia.org/w/index.php?title=SAP_SE&oldid=872245407

ScWi18. (2018, 10 30). *Wikipedia - Scratch*. Retrieved 12 10, 2018, from https://en.wikipedia.org/w/index.php?title=Scratch_(programming_language)&oldid=866532637

ShWi18. (2018, 08 28). *Wikipedia - Shareware*. Retrieved 09 10, 2018, from Wikipedia: https://en.wikipedia.org/w/index.php?title=Shareware&oldid=846878603

SnWi18. (2018, 09 28). *Wikipedia - Snap! (programming language)*. Retrieved 12 10, 2018, from Wikipedia: https://en.wikipedia.org/w/index.php?title=Snap!_(programming_language)&oldid=861562040

SoWi18. (2018). *Wikipedia - Public-Domain Software*. Retrieved 12 11, 2018, from https://en.wikipedia.org/w/index.php?title=Public-domain_software&oldid=869332992

SuTh18. (2018). Ahmed Ibrahim, Craig Vallilan McAteer, Junaid Chaudhry. *The Journal of - A security review of local government using NIST CSF, 74*(10), pp 5171–5186.

SuWi18. (2018). *Wikipedia - SUN Microsystems*. Retrieved 12 16, 2018, from https://en.wikipedia.org/w/index.php?title=Sun_Microsystems&oldid=873668765

TeGo17. (2017). *Kunal Parikh - why Google open-sourced TensorFlow*. Retrieved 11 21, 2018, from https://hub.packtpub.com/google-opensorced-tensorflow/

TeHu18. (2018). *TensorFlow Packhub*. Retrieved 11 15, 2018, from https://hub.packtpub.com/google-opensorced-tensorflow/

TesW18. (2018). *Tesla Inc, - Wikipedia*. Retrieved 11 15, 2018, from https://en.wikipedia.org/w/index.php?title=Tesla,_Inc.&oldid=868708607

TeWi18. (2018). *Wikipedia - TensorFlow*. Retrieved 12 16, 2018, from https://en.wikipedia.org/w/index.php?title=TensorFlow&oldid=871761161

UiGi14. (2014). *GitHub - SAP OpenUI5*. Retrieved 05 10, 2019, from https://github.com/SAP/openui5

UnSc15. (2015, 10 9). *ScienceDirect - Understanding Enterprise Open Source Software Evolution.* Retrieved 04 24, 2019, from https://pdf.sciencedirectassets.com/280203/1-s2.0-S1877050915X00251/1-s2.0-S1877050915027441/main.pdf?x-amz-security-token=AgoJb3JpZ2luX2VjEPr%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXV

zLWVhc3QtMSJIMEYCIQC8HtrYdigI4SbPVxG5C0D51Ar%2BMlo8qvEH%2FS37YO
Y1gQIhAPOYBYBZ

UnWi18. (2018). *UNIX Wikipedia*. Retrieved 11 03, 2018, from https://en.wikipedia.org/w/index.php?title=Unix&oldid=864040442

UpsSa18. (2018). *news.sap.com - New SAP Upscale Commerce Solution Extends Customer Experience Ecosystem with Open Integration Tools*. Retrieved 11 23, 2018, from https://news.sap.com/2018/10/sap-upscale-commerce-extends-customer-experience-ecosystem-open-integration-tools/

VaWi18. (2018). *Wikipedia - Value Network*. Retrieved 12 16, 2018, from https://en.wikipedia.org/w/index.php?title=Value_network&oldid=865014546

VenWi18. (2018). *en.wikipedia.org - Vendor Lock-in*. Retrieved 11 23, 2018, from https://en.wikipedia.org/w/index.php?title=Vendor_lock-in&oldid=870220812

WaIn08. (2008). InfoWorld.com - What Cloud Computing Really Means - Eric Knorr, Galen Gruman. *http://skysolutions.co.zw/docs/What_Cloud_Computing_Really_Means.pdf, 14*(13), 3.

WhIM12. (2012). Why do commercial companies contribute to open source software? *International Journal of Information Management, 32*, 106-117.

WhOp16. (2016). *What is the cloud - opensource.com*. Retrieved 11 23, 2018, from https://opensource.com/resources/cloud

WiAn06. (2006). *Anthony D. Williams - Wikinomics: How Mass Collaboration Changes Everything.* US: Tantor Media.

WinWi18. (2018). *en.wikipedia.org - Winner-take-all market*. Retrieved 11 23, 2018, from https://en.wikipedia.org/w/index.php?title=Winner-take-all_market&oldid=859399550

WiVM18. (2018). *Wikipedia - VMware*. Retrieved 12 09, 2018, from https://en.wikipedia.org/w/index.php?title=VMware&oldid=872748453

WMU18. (https://www.unido.org/sites/default/files/files/2018-06/World_manufacturing_production_2018_q1.pdf). *World Manufacturing Production.* United Nations Industrial Development Organization.