AIUB
Office of
Research and Publications

Exploration of open source licensing model as a tool to enhance digitalization in Bangladesh

Raihan Kibria
Md. Khalid Rahman

AIUB Journal of Business and Economics
Volume: 17 Issue Number: 1 ISSN (Online): 2706-7076

November 2020

Citation

Kibria, R & Rahman, M.K. (2020). Exploration of open source licensing model as a tool to enhance digitalization in Bangladesh. *AIUB Journal of Business and Economics*, 17 (1), 129-158.

AIUB Journal of Business and Economics

Volume 17, Issue 1

ISSN (PRINT) 1683-8742

ISSN (ONLINE) 2706-7076

November 2020 Page 129-158

Exploration of open source licensing model as a tool to enhance digitalization in Bangladesh

Raihan Kibria*

Department of MIS

Faculty of Business Administration

American International University-Bangladesh (AIUB)

Md. Khalid Rahman

Department of Law

Faculty of Arts and Social Sciences

American International University-Bangladesh (AIUB)

Corresponding author*: Raihan Kibria
Email: raihan.kibria@aiub.edu

Exploration of open source licensing model as a tool to enhance digitalization in Bangladesh

Abstract

The twenty first century offers us the greatest gift that human kind can achieve – technology by which we can now explore the riches of global village from any corner of the world. In the global village that we live in, information technology has made tremendous strides to make the lives of people more productive and more convenient. Bangladesh has been making inroads into the adoption of technology though at a slower rate than the developed world. In this paper, we have explored socio-legal framework that would facilitate digitalization and benefit for the stakeholders, namely, consumers, vendors, software developers and society in general. In particular, we have explored the modalities of open source licensing and investigated the barriers, its positive impacts on consumers and developers, and implementation of the licensing model from social and legal perspectives. It is a general perception that the cost of licensing remains out of reach for most of the computer users in Bangladesh. Most of the popular software are proprietary in nature, copyright protected and of foreign origin. As a result of prohibiting cost of acquiring software, consumers are often dissuaded from purchasing licensed products. As a consequence, consumers in the developing countries such as Bangladesh are inclined to use non-licensed software, which is a clear violation of copyright laws, and suffers from lacking features and support for the software. On the contrary, open source licensing holds the promise of delivering software free of cost with creative features and appropriate customization for the end-users. We conducted an exploratory study on the implications of using open-source licensing both in legal and societal contexts of Bangladesh. We relied mostly on secondary data. Our research demonstrates that open source licensing could facilitate the adoption of technology at a higher rate. Open source licensing could expedite the use and distribution of software in conformance with national and international legal standards. We also found that compared to developed countries, Bangladeshi users are not often encouraged from the government bodies to use and disseminate open-source licensed products.

Keywords: FOSS, Software, Open-source, Copyright, WIPO, Digitalization, Unlicensed Software.

1. Introduction

Owing to the advent of the Internet and wide proliferation of digital technologies, businesses have evolved and newer methods of value generating activities have emerged. Software business is no exception. Traditionally, software companies develop the product under their tutelage with a group of developers. The developer company manages the distribution rights, copyrights, and generates revenue by selling copies of the software. This business model, over the last few decades, have been complemented and, in some cases, challenged by open-source development, where groups of developers join to develop software and give access to the general public to use the product for free.

Free and Open Source Software (FOSS) is a practice of writing computer programs and releasing the output, both binary and source code, under a license that permits others to modify product. In recent years, FOSS has developed into a new model of collaborative endeavor. The end-users get benefit from the software for free of charges. The commercial enterprises derive benefits from having contributors from all over the world. The free-flow of ideas and intellectual knowledge putatively foster a collaborative model that enhances innovation. To highlight how far open-source movement have come to be one need not look farther the recent pandemic of COVID-19. Finding a cure requires researchers to pore through thousands of relevant research papers. Making the findings easily searchable has been a key goal. To accomplish this, many people around the world are collaborating in reputed sites such as Kaggle, nlpcovid19workshop.com. Pasteur Institute and Greater Paris University Hospitals released their auto-diagnostic bots under open-source license.

Open source has been in the mainstream ever since Linux operating system had been successfully developed by developers around the world in early 1990s. The smart phone revolution in the last decade has been aided by adopting a version of Linux by Google Inc. to use as Android mobile phone's operating system. An ordinary user may not have noticed, but the phone already bears the labors of thousands of volunteers of open-source community.

We present the modalities of open source licensing and investigate the barriers, its positive impacts on users and developers, and implementation of the licensing model from a social and legal perspective. We have highlighted existing copyright laws both at national and international levels. To put the open-source licensing in the broader context, we have also

demonstrated existing open-source software, list of statistics on the use of unlicensed software, and litigations that arose out of open-source licensing models.

1.1. Problem Statement

Releasing source code and granting distribution rights of programs to community seem counter-intuitive from profit-making perspective. After all, why would a developer dole out their labor and innovation for free? Our study addresses this question by finding out underlying motivating factors that inspires a contributor. The other question we seek to answer is how society benefits from open-source model. In this regard we review the cynic's view that companies and all human enterprise strives to maximize profit. We have explored how, if at all, a community-driven free-for-all open-source model has any place in the market economy. Even if this method of software development does have a place, why a developing country like Bangladesh is lacking in adopting open-source? Do we have enough public policy or do we need one to encourage open-source model? These are the questions we seek to answer in our study.

1.2. Literature Review

It is worth exploring what the term "open-source" means. Many people mistakenly use the terms "free software" and "open-source software" interchangeably. A free software, just like any free goods, is something that the maker gives without a fee. The maker of the software has not given the source code; rather just the binaries. Free software, like a slew of anti-virus software and Internet browsers, comes as "as-is" basis, unable to be modified. On the other hand, an open-source software makes its source code available to the users with the know-how, modify the source code benefitting for their requirements. While most users are not needed to change the source code, the developer community makes use of the code to add or modify functionalities. Lauren (2003) identifies that the eligibility of open-source software is often weighed against few criteria. Most salient is granting the user free distribution right, which is freedom for distributing the product and source code. Another major criterion is allowing source-code to the grantee for modifying and passing on the changes to the code to the public. Not only the non-profit user/ organizaion can use/distribute the software, but also the profit-making corporates are granted equal rights. In fact, several open-source projects have got a mix of community developers' contribution and corporate patented algorithms. This gives rise to intellectual property issues. In view of

complications regarding distributions, various licensing schemes are associated with a project. An open-source project is accompanied with a Contributor's Agreement (CA), which binds the users and developers to terms that are defined at the initiation of the open-source project (Guadamuz & Rens, 2013). Some Open Source Licensing (OSL) schemes, allow the companies to charge users from the changes company made. Some OSL scheme requires that any changes be made public. Some licensing schemes are more restrictive in not allowing companies to charge users.

Given differing needs of developers and legacy of open-source software, there is a need to certify what is open-source and what is not. Software, to be qualified an open-source license, must be approved by the Open Source Initiative's license review process (OSI, 2020), a process meant to ensure that the software bear license that allows the software to be freely used, changed and shared. Since the source-code may be modified in future, the original developer may leave a license statement in the source-code. This license statement could also be an individual file. Subsequent developers or distributors may be required to obey the terms delineated in the license and pass on the original license to subsequent developers or distributor down the chain. This creative style of licensing, called copyleft license, ensures that the spirit of open-source is preserved in subsequent releases (European Parliament, 2013).

Studies show that number of open-source projects has grown almost exponentially (Deshpande & Riehle, 2008). Various legal issues have cropped up (Rajiv, 2018). Does society and developers really benefit from non-profit making open-source initiatives is a question some researchers try to answer (Hippel & Krogh, 2003). To answer the status of software licensing we reviewed a few surveys (BSA, 2018). To understand the dynamics of development in the early stages of computer software history, we reviewed research papers and existing copyright related international.

1.3. Objective of the Study

The objective of the study is to explore the factors that make open-source development sustainable in a developing country like Bangladesh. To understand the market and community dynamics of software development, we broke down the general objective stated above into the following specific objectives:

I. Evaluate status of open-source projects world-wide

II. Explore the financial sustainability of open-source endeavours in capitalistic profit-making context
III. Highlight legal issues and court cases arose from intellectual property right
IV. Identify how policy makers in Bangladesh can benefit from global trend

2. Methodology

The research method is exploratory. Since open-source methodology is novel, established literature base is rather scant. We had to find novel sources of research information. To explore various licensing schemes, we downloaded several reputed open-source products and perused their licensing terms. These licensing terms and conditions are available as text files. We also incorporated statistics gathered from reputed organizations, such as Business Software Alliance (BSA, 2018). In addition, we consulted policies of established open-source sponsoring organizations, such as Open Source Initiative. As for the traditional type of information gathering, we reviewed published research papers both technical and legal, text books and professional journals available globally for the last two decades.

3. Analyses and findings

Subject as novel as open-source development requires a review of history of early computers and software development. We based our analysis as such taking into context the historical practices, technical evolution and gradual commercialization of software.

3.1. History of open-source software

We first explored the structure of the open-source project in terms of people involved. Then we gave a summary account of the open-source movement.

3.1.1. Open source project structure

A typical open-source project consists of a number of contributors and a maintainer as shown in Figure 1. Generally, the contributors are computer professionals from a wide variety of computing discipline. A vast majority of them are developers and computer scientists. In addition, there are some professionals who document the system and maintain project wiki and project collaboration tools. The job of the maintainer is to decide what contribution to accepts, what to reject and what to revise. In addition, the maintainer coordinates documentation, testing strategy and release planning.

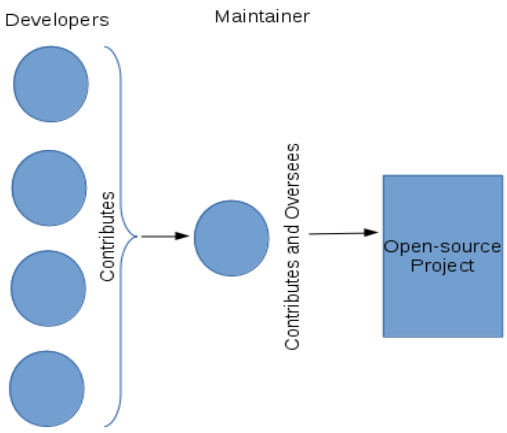

Figure 1: Typical structure of open-source projects

3.1.2. The early days of sharing computer software

The practice of sharing code started as early as 1950 during the beginning of computing in the modern era of digital electronics (Scacchi, 2007). During those days' hardware was the main driving force in the electronics business; little revenue was generated from software. For example, IBM launched its first digital computer model 701 in 1953 and subsequently launched the successor product model 704 in 1954 (Campbell & Garcia, 2009). Using the hardware required writing programs, an endeavor IBM spent little effort on. As a result, the users of IBM 701 and 704 users had to write software on their own. This resulted in about $150,000 expense which is equivalent to the first year rental for the hardware. The lack of common programming infrastructure also resulted in duplication of efforts as for similar functionality users wrote their own programs in isolation. IBM facilitated creation of a user forum SHARE that promoted sharing of software among the users of IBM digital computers. IBM continued supporting SHARE for the two following decades.

As rapid progress came to being in computer hardware and integrated circuit technology, the academic community took interest in developing algorithms, languages and tools to develop new and capability enriched systems. Software as a separate stream of revenue was still deemed not viable by the computer manufacturers; it was the computer researchers, academicians and scientific community who studied and wrote programs such as compilers/interpreters and databases. During that time, personal computers and software were not as widespread as it is now. Researchers' main efforts

were directed towards writing programs to automate businesses and to solve scientific problems. Early computer languages such as Pascal, C were being conceived. Various compilers and interpreters, among other tools, were developed. The trend of sharing code and know-how in 1970-1980 was set and advanced in full steam as researchers not only published their know-how but also code. It can be evidenced in the published books during this time-frame (Press et al., 1986).

Until 1980s, there was a dearth of collaborative tool to share and modify the code, document the logic and track versions of the software. A major project was undertaken at the University of Southern California, where programs developed by many contributors, especially graduate students, were tracked and managed by using an experimental and evolving software collaboration tool called The USC System Factory Project (Scacchi, 1989). The collaboration tool provided a veritable testing ground for issues involving large-scale software development, where many programmers had to be coordinated and code management became a large issue during staff change-over. This project also brought to life developer friendly hypertext-enabled documentation bringing ease to the collaborative development process. Although the project was not used in a massive scale as many open-source projects are now, the project was a pioneering effort in automating sharing of code, knowledge and harboring creativity that many open-source management tool later came to adopting.

During the 1990s, when the Internet enabled users all over the world to get connected, sharing work and programming code gathered momentum. It is this time which is identified most with the inception of FOSS. One of the most popularly cited open-source project GNU Linux was initiated in 1991 (Hars, 2002). Subsequently in 1995, Apache, a major web server used in the world today, was built by developers located throughout the world.

3.1.3. Empirical studies on the number of open-source projects

During the early period of computers and software development in the middle of the last century, the number of contributors were a select few personnel from academia and scientific community. Other than to a privileged few enthusiasts, the infrastructure to share code and knowledge were limited. However, that changed after the dawn of the Internet. Since the early 1990s there has been exponential availability of digital resources and access to code. We can see from Figure 2 and Figure 3 that there have been a rapid increase in the number of FOSS projects and also the number of lines of programs shared by contributors of open-source project (Deshpande & Riehle, 2008).

The contributors are estimated to be from countries which number well-above 200. A rapidly developing country such as Bangladesh also hosts a substantial number of university graduates who contribute to the global open-source communities.

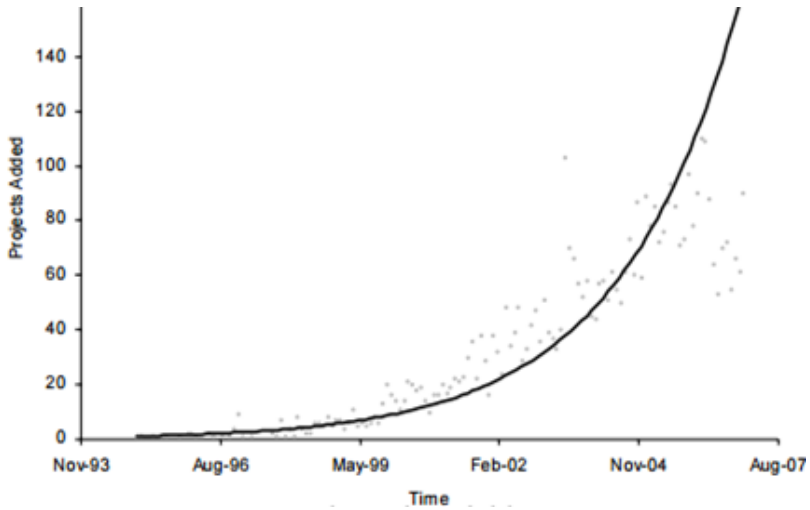

Figure 2: Number of open-source projects added
Source: Deschpande Riehle, 2018

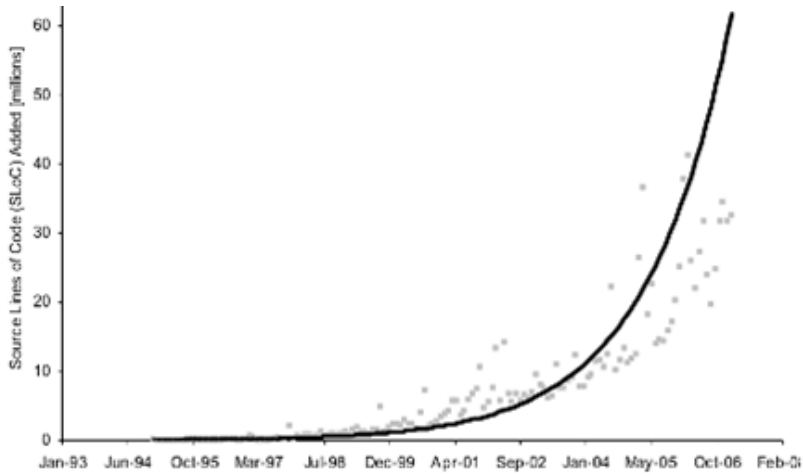

Figure 3: Number of lines of codes
Source: Deschpande Riehle, 2018

3.2. Motivations for contributors of open-source software

3.2.1. Why contributors share their knowledge and programming code

A number of studies have been done to identify the reason why developers share the knowledge and programs with no apparent benefit. The community wondered why a product such as GNU Linux or Apache would succeed and maintain the quality and the number of developers that contributed to develop them. We have summarized the contributors' benefit in Figure 4. During the early days of sharing code starting in the 50s were limited to a few scientists, in the latter years beginning 1990s sharing code involved thousands of developers. While the early open-source community was driven to share knowledge for the sake of a brethren ship and camaraderie in community, the contributors of the later days did not possibly know each other. Studies indicate that several personal motives are at force. (Maxwell, 2006). Some of the key motives are:

i) Altruism and conformance to the norm among peers developers

ii) Intrinsic benefit of joy of problem solving

iii) Solving a personal problem

iv) Recognition by professional network and peers

v) Self-improvement though skill enhancement

vi) Reduced barrier to collaborate

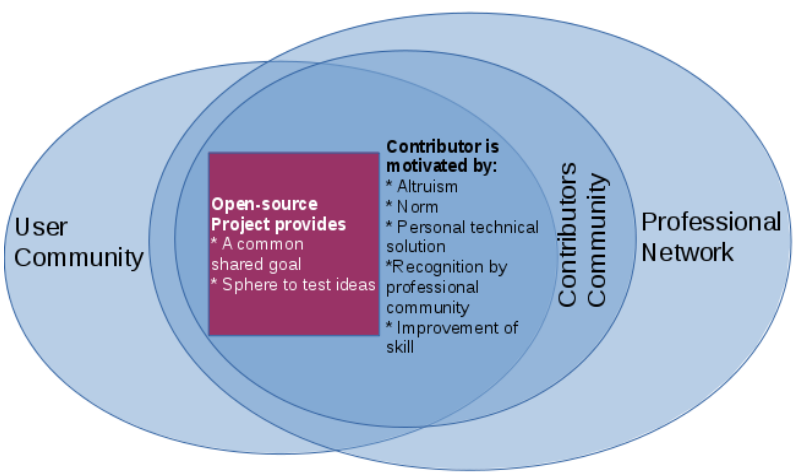

Figure 4: Open-source contributor's motive paradigm

3.2.2. Altruism motive to share solution and conformance to the norm among peers developers

Benevolence is a driving force for sharing know-how. Given the history of sharing code in the early days of programming this is not a surprise that many open-source contributors admit that altruism spurs them to share their code and solutions. When people feel that they are part of a larger community, the motivation for publishing and sharing their work achieves a new height. (Maxwell, 2006).

The practice of sharing programs has its root in the SHARE user group of IBM and DEC user-group. The lack of sufficient programs to solve real-life problems motivated the programmers and academic community to share code. This was time when software and hardware came as bundled end-product giving users no choice to rely on third-party software Developer Company. If the user is not satisfied with the software that came bundled, it was up to him or her to write his or her own program. Software as a separate product and a source of revenue was yet to be ideated. Programmers shared their solution with peers to have others of what worked for them.

The trend of sharing continued later on culminating in the creation of FSF (Free Software Foundation) by Richard Stallman in the 80s. One of the goals of the FSF is to promote free software, an altruistic motive zealously supported by ardent supporters. It is well-known that the natural tendency of a human being is to be helpful: as a social creature we long for belonging and being of help to others. Sharing our know-how on a problem and serving others with the solution/program serve the altruistic motive.

Through the work an individual tries to be of benefit to others – this means that a person will often forgo a part of their apparent material gain in order to see that what he or she contributes serves a bigger purpose. Many of us know someone who rejected monetarily high-paying jobs in favor of a lower-paying work that helps him or her serve community better.

3.2.3. Intrinsic benefit of joy of problem solving

Open-source projects open doors to the programmers and problem solvers to try out their own ideas on a problem that is open to public. The fact that the problem statements are often publicly available helps developers get a quick understanding of the problem and analyze attempted techniques, which facilitates trying out their own take on the issue. Developers like any other creative people derive the joy of creativity and innate satisfaction just from arriving at a solution; finding his or her own way in the maze of unknowns

and seeing the light at the end of the tunnel provides enough motivation to tackle unsolved problems. That the solution developer improvised may be reviewed by thousands or accepted by the community provides added spur and promises to boost his or her confidence and self-belief. These intrinsic rewards can never be overstated.

3.2.4. Solving a personal problem

Many at times software developed in open-source arena are downloaded and tried by professionals/academicians/researchers to fit their own specific needs. Sometimes, the user finds that some features are lacking, need improvements or need some tweaking to perfectly solve the purpose in hand. This is when the user himself/herself may modify the source code. By his/her coding, the user may add a new feature, provide improvements to the existing code or devise an alternative way. Upon committing the changes to the original open-source site, the user helps to enrich the product.

As an example, the dominant web server Apache has its roots in the early days of the Internet when there was a real void in feature-rich web server that would help organizations serve millions of people. Developers often downloaded the bare minimum Apache web server software from the Internet, added a module that needed to be added and contributed the module/feature to the site. Through many such modifications and additions Apache became feature-full and robust to be able to boast itself as one of the most popular web server this days. (Friedman, 2006).

Open-source projects open doors to the programmers and problem solvers to try out their own ideas on a problem that is open to public. For some problems, there may be per-existing code fragments, documentations or partial solutions posted on the public domain by other problem-solver who attempted the issue previously. For some problems, the problem statement may be altogether new. In either case, an aspirant problem-solver is motivated to tackle the problem and often finds some pertinent resources. Once on to the problem, programmers like any other creative people derive the joy of creativity and innate satisfaction. Just like a poet or an artist, who often work on unpublished work just for the sake of creativity, programmers find the urge to provide solution to problem in a unique way.

3.2.5. Recognition by professional network and peers

One key aspect which became apparent as open-source projects became mainstream is the acceptance of the contributors by the peers. It is a well-

known fact that community which includes large tech giants often watches out for good performers whose works have substantially enriched a product. Owing to the ubiquity of the Internet, words spread quickly to bring a star contributor to the attention of peers and professional community. This, in turn, improves the status of the contributor giving him or her more clout, resource and say on things – developers who are well-recognized are often rewarded with the controlling authority in steering the direction of the community endeavor. The very nature of collaborative development fosters meritocracy, where a contributor is evaluated based on his or her contribution – being useful in the project elevates the programmers' status in the community (Lerner & Tirole, 2003).

3.2.6. Self-improvement though skill enhancement

A key reason, which is often overlooked, is that a developer or professional is in the race with time and peers. A contributor often finds himself in need of new challenges and avenues where he or she can practice and hone his or her skills. Open-source projects provide a perfect ground for such professionals or students to try out their skills. As an enticement, the work may be accepted by the community providing him or her with credence to display to potential recruiter or a would-be supervisor.

3.2.7. Reduced barrier to collaborate

Owing to increased availability of quality Internet connection and better means to collaborate online, a vast number of users/developers now has a veritable mean to share and contribute. Code sharing and project management tools, such as GitHub, Source forge, facilitate development and documentation with an ease that is unprecedented anytime in the modern history. The falling price of storage and communication devices also led to affordable and sometimes-chargeless infrastructure that is a prerequisite for freely sharing information and know-how. Together with increased skill level, these improvements in infrastructure serve as complementary assets that inspire the contributors to share more and gain more from open-source projects. (Weber, 2004).

Improvement in software development methodology also facilitated collaboration. Open-source projects tend to follow a design model that is modular in structure and follow basic principle of service-oriented-design whereby peripheral clients can use API calls to the core engine making it unnecessary for the developer to learn the project in its entirety but still be able to modify only a part of the product to get the job done (Weber, 2004).

Modular architecture of the projects is open to accepting more changes to the product as the user changes are tracked and published openly resulting in more non-conflicting commits.

3.3. Open-source license schemes in practice today

Open-source license schemes are geared towards maximizing developer's contribution and keeping the product free for use. There are many different sub-schemes allowing and restricting users and developers to distribute the software in varying degrees. Below are the most popular schemes active currently (Lerner & Tirole, 2003).

3.3.1. GPL version 3 License

One of the most popular licenses GPL (General Public License) has its roots in the 80's. Linux operating system, GNU compiler collections are some of the notable software holding this license scheme. It is a copyleft license where each modifier and beneficiary of the software must receive and propagate the license that the original developer had embedded with the software. The goal of the copyleft scheme is to sustain the original purpose of the software, which is often to be able to freely modify and distribute and yet keep the derivative work free to use. The source code for any derivative work must be made public and free under this license. The success of the GPL is evident today as Linux and many software released under this license are flourishing today benefiting a wide variety of users. One of the objectives of this license is to make license viral – by requiring the original copyleft license with the derivative works the self-perpetuating license scheme propagates. By requiring that original license text must be borne with the derivative works, reciprocity is achieved – the beneficiary of a software ensures that future beneficiaries from the modification/addition get the same benefits.

Success of any open-source software licensing scheme is contingent on its ability to work within the perimeters of existing copyright and patent laws, which are broader in their coverage of intellectual work encompassing other types of work, such as art, mechanical/engineering invention and trade secrets. GPL scheme has evolved over the years to navigate the ever-changing terrains of intellectual property rights, resulting in version 3 in 2007. GPL v3 has been written keeping in mind certain international statutes such as the Digital Millennium Copyright Act and the European Union Copyright Directive facilitating usage of this license among international developers. The license also prevents large corporations from unintended infringing on patent rights, royalty collection.

3.3.2. MIT License

This license if more permissive than GPL license: it allows users to freely use and modify the software without imposing the copyleft restriction present in GPL thus providing rights to freely distribute the software to the extent that the distributor can reap commercial gain without having to mention the original license. The license makes it clear that the software is provided "as-is" basis without any obligations or liabilities on the developer. By not requiring copyleft license, unlike GPL, MIT license does not lend itself to be viral.

3.3.3. Apache 2.0 License

A very popular license which boasts several successful open-source projects such as the Apache Web-server, this license is more permissive than GPL in terms of distribution rights and monetizing derivative works. The license does not require any derivative works to be released under the original license. Earlier versions of the license came to being in the late 1990's and early 2000's. The later version the Apache 2.0 license has been written keeping in mind excessive litigations related to patents that came to being in the later decades. Firstly, all a contributor must include patent licenses in the software. Thus the users are entitled to use the software without violating patents. Secondly, if the user files legal complaints against patents, his/her license is discontinued. The license scheme is compatible with GPL v3 license with one caveat: the work that contains both Apache and GPL v3 licenses, must be released under GPL v3. By ensuring that user is free to use/distribute the software without having to worry about patent or copyright violation, contribution to software is greatly increased.

3.4. Current notable open-source software

This section gives an overview of some open-source projects that are used widely. Instead of being comprehensive, we try to highlight the success of the open-source projects listed below.

3.4.1. Linux

One of the most popular operating systems, Linux serves a sizable number of web users today. Since its inception and distribution under GPL in the early 90's, Linux has become a model and a success story of open-source movement. Through the years it has proven to the world that a free software can become feature rich and adaptable in modern world. Google uses Linux

kernel in it widely popular Android operating system. In the early days, Linux proved its worth in the domain of servers; later, it has proven useful in user-friendly user interface resulting in desktop operating system such as Ubuntu.

3.4.2. Ubuntu

Ubuntu, the desktop version of Linux, provides users with graphical interface, which is something similar to existing commercial desktop operating systems. Through easy to use features such as friendly software installation, networking interface, interoperability with commercial software, Ubuntu has become a viable desktop alternative. Various open-source packages, such as LibreOffice, GIMP, if not pre-installed can be easily installed on Ubuntu, giving the users comfort of use that parallels commercially available software.

3.4.3. MySQL

MySQL is one of the most widely used databases available since 1995. Through the cooperation of a thriving community of enthusiastic developers, the database became robust, feature-rich, scalable and compliant to standards. Notable users include Facebook, YouTube, Twitter and NASA. Compared with commercial counterparts, MySQL server remains easy to install prompting a vast number of downloads and use by the developers and students. The algorithms developed to retrieve and store data remains state-of-the-art.

3.4.4. Apache web server

Apache web server, released in the late nineties, became one of the most dominant web servers. It has become the first web servers to serve more than 100 million websites. Early on, its adoption of modular architecture lent itself to quick addition and modification of features. User are free to configure the server as they require by enabling and disabling modules. The web server now boasts features that are unparalleled by its contemporaries. The server over the years has become robust in handling huge load, tamper proof in terms of security and versatile in handling multiple domains.

3.4.5. Firefox web browser

One of the most popular browsers and one of the most open, Firefox was launched in 2004. Its usage grew to a peak of 32% at the end of 2009. As other commercial browsers came into the market, user privacy issues started to be discussed in the public arena. Over the years, the browser evolved into

more friendly, yet it remained open in terms of how it handles privacy issues. It also rewards users who points out bugs in its security (Mozilla, 2020).

3.4.6. WordPress

One of the most popular content management system, WordPress, started its journey in 2003. As the number of bloggers grew, the Internet saw a need for a blog software that is easy to use and publish with. Through its plugin architecture, WordPress allows customization as the user sees fit. Currently, about a quarter of blog sites runs by WordPress (Mirdha et al., 2015).

3.5. Legal Paradigm/framework on both national and international level

3.5.1. Legal protection to software

As a human expression of novelty, software is relatively new. As to what form of legal protection was to give to software various options were open for debate, copyright being the first, existing copyright statutes was universally accepted under Berne Convention, which will be shortly described. The covenant establishes a world-wide Copyright Union that automatically and uniformly protects such work (European Parliament, 2013). In Europe, copyright related to software were finalized under Software Directive (Directive 91/250/EEC, 1991).

3.5.2. Software Protection: International Instruments and Trends

3.5.2.1 TRIPS

TRIPS Agreement explicitly includes computer software in its list of copyrighted works. The agreement mandates, in Article 10, that its member states protect software as literary works under the Berne Convention. Specifically, the source code and binaries are subject to this copyright protection. However, the efficiency of protection offered by copyright is adversely affected by the idea expression dichotomy. Algorithms which, are just ideas, are thus excluded from this protection (Mahapatra, 2011).

3.5.2.2 Berne convention

Berne convention became effective on 5th December 1887, and forms the basis of most international instruments for the protection of intellectual property rights. Unlike TRIPS, Berne convention does not explicitly extend the protection of copyright to software. However, it should be kept in mind is

that the term Works enumerated in Article 2 of Berne Convention is not exhaustive. Rather Article 2 (7) of the Berne Convention makes the protection of works contingent on the statute of the originating country (Mathan, 2014).

3.5.2.3 WIPO Copyright Treaty

The WIPO Copyright Treaty (WCT) extends Berne Convention regarding copyright to digital environment. Article 4 of the Treaty expressly states that, "Computer programs are protected as literary works within the meaning of Article 2 of the Berne Convention. Such protection applies to computer programs, whatever may be the mode or form of their expression" (WIPO, 1996). Article 6-8 grants the creator the right of distribution, right of rental, right of communication to the public (Sudha, 2020).

3.6. Some open-source software related litigations

In this section we have introduced readers to some recent litigations involving licenses of open-source projects. We have tried to highlight the nature of litigation that could arise from misuse of open-source policies.

3.6.1. CoKinetic Systems Corporation vs. Panasonic Avionics Corporation

Software bearing GPL requires that any modifications to it be kept public and published under the same license as the original (Reciprocity). However, CoKnentic Corporation, in a March, 2017 lawsuit claims that Panasonic Avionic Corporation violated the reciprocity clause when the latter refused to release source code of its modified version of the Linux operating system that it used for in-flight entertainment. Panasonic being a global player in the industry, thus infringed upon the license and prevented competitors from developing complementary software such as device drivers, CoKinetic alleges (Rajiv, 2018).

3.6.2. Oracle vs. Google: Java API case

Java, one of the most popular modern computer languages that is executable on all operating systems, was originally developed and maintained by Sun in early 1990s. Beginning of 2005, Google desired to include Java in its Android operating system. Instead of developing on the Sun's Java code base, Google strayed and developed its own version. However, some of the APIs (Application Programming Interface) that is used for inter-operation and managing calls from other modules, remained the same. This prompted a lawsuit by Oracle, which by now is owned Sun and Java, alleging intellectual

property violation by Google. The case was eventually dismissed on the ground that APIs are not copyrightable. (Oracle America, Inc. v. Google, Inc., 2018)

3.7. Statistics on unlicensed software at national and international level

Software piracy has become a major problem in the field of technological advancement. Today, software privacy is an issue of global importance. With high speed internet connectivity and increased power of computing technologies, there has been an increasing trend toward software piracy. Software companies have been affected by piracy for years. BSA's Global Software Survey and IDC made a study to gauge the extent to which personal computers of more than 100 economies are hosting pirated software (BSA, 2018).

The study shows that 37 percent of the software being used does not have proper license (Figure 5). Also, use of unlicensed software can cost a company more than ten thousand dollars per infected computer and costs IT divisions of all companies globally nearly $359 billion a year (BSA, 2018). IDC estimated use of unlicensed software on average exposes a computer to 29 percent malware infection (BSA, 2018).

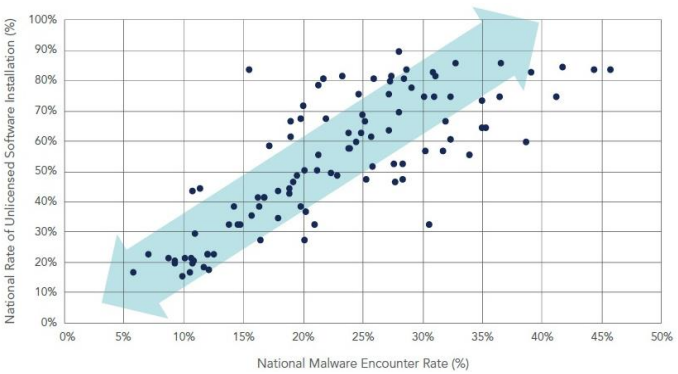

Figure 5: Unlicensed Software and Malware Encounters Are Tightly Linked
Source: BSA, 2018

The study also shows that there has been a little decrease in pirated software use, the world-wide rate dropping two percent points from 2015 to 2017. It is also evident from Figure 6 that emerging markets have higher than normal unlicensed software usage rate (BSA, 2018).

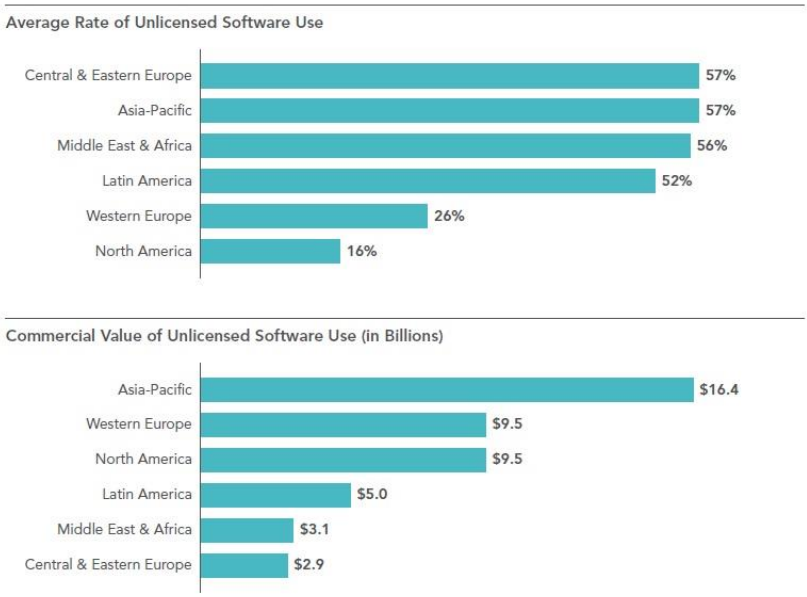

Figure 6: Unlicensed software statistics world-wide
Source: BSA, 2018

In Bangladesh, the scenario of using unlicensed software is a alarming. Though the percentage of usage has been declining from 86% to 84%, it is still the highest number in the Asia Pacific region (Figure 7).

RATES AND COMMERCIAL VALUES OF UNLICENSED PC SOFTWARE INSTALLATIONS

	RATES OF UNLICENSED SOFTWARE INSTALLATION				COMMERCIAL VALUE OF UNLICENSED SOFTWARE ($M)			
	2017	2015	2013	2011	2017	2015	2013	2011
ASIA PACIFIC								
Australia	18%	20%	21%	23%	$540	$579	$743	$763
Bangladesh	84%	86%	87%	90%	$226	$236	$197	$147
Brunei	64%	66%	66%	67%	$18	$19	$13	$25
China	66%	70%	74%	77%	$6,842	$8,657	$8,767	$8,902
Hong Kong	38%	41%	43%	43%	$277	$320	$316	$232
India	56%	58%	60%	63%	$2,474	$2,684	$2,911	$2,930
Indonesia	83%	84%	84%	86%	$1,095	$1,145	$1,463	$1,467
Japan	16%	18%	19%	21%	$982	$994	$1,349	$1,875
Malaysia	51%	53%	54%	55%	$395	$456	$616	$657
New Zealand	16%	18%	20%	22%	$62	$66	$78	$99
Pakistan	83%	84%	85%	86%	$267	$276	$344	$278
Philippines	64%	67%	69%	70%	$388	$431	$444	$338
Singapore	27%	30%	32%	33%	$235	$290	$344	$255
South Korea	32%	35%	38%	40%	$598	$657	$712	$815
Sri Lanka	77%	79%	83%	84%	$138	$163	$187	$86
Taiwan	34%	36%	38%	37%	$254	$264	$305	$293
Thailand	66%	69%	71%	72%	$714	$738	$869	$852
Vietnam	74%	78%	81%	81%	$492	$598	$620	$395
Other AP	87%	87%	91%	91%	$442	$491	$763	$589
TOTAL AP	**57%**	**61%**	**62%**	**60%**	**$16,439**	**$19,064**	**$21,041**	**$20,998**

Figure 7: Unlicensed software statistics by county
Source: BSA, 2018

3.8. Accelerating digitalization by adopting FOSS nationally and internationally

3.8.1. How OSL helps digitalization and issues facing in current world

In this section we show how OSL helps to adopt digitalization at a faster pace. In addition, we also delineate some issues facing open-source adoptions. Open-source benefits the users because open-source software is free by definition and the development community intends to keep it open. The license schemes are protective of the very right that software be free to use. We see that a great number of people use Mozilla Firefox as a software in the consumer-end. We also see that unbeknownst to an end-user the very website he or she visits may rely on open-source server such as Apache to serve pages. There is indeed no dearth of evidence showing that open-source software helps more adoption to digitalization.

3.8.2. Benefits to the end-user – in the near and long-term

Computer users benefit from using a stable and quality software. Open-source software, by virtue of its being open, is available people all over the world. As a result of tremendous number of downloads of the software and repeated use by a great variety of users, the software gradually becomes feature-rich and robust. It parallels the proprietary software and sometimes out-competes the proprietary software in terms of bug-free-ness. Corporates typically house a testing team for ensuring quality of the product. In FOSS, many people use the product and implicitly become testers. A well-managed open-source software project thus becomes not only a free to use, but also a quality product.

3.8.3. Benefits to the corporate

Corporate benefits from the open-source movement in many ways – some subtle and some obvious. One of the indirect benefits is corporates can reap the fruits of labors of many contributors, their fruit of labor being a quality, robust and feature-rich package. Since the pool of contributor is not limited by geography or national boundary, the pool of talented programmers and their skills are over time almost endless. This collaboration often results in a mature product in a shorter span of time than would be if corporates took control of the product alone. Corporates often reap some obvious benefits such as selling auxiliary services or complementary products related to the main open-source products. Also by collaborating with the open-source community, corporates have easy access to potential recruits who have been

proven to be a major contributor. Corporates also have easy access to programming breakthrough, insights into the end-users as open-source projects are open to the corporates as well.

3.8.4. Benefits to the society

We have shown in section 2.6 that pace of innovation is higher in a private-collective model than in either private model or collective model alone. Society benefits from increased speed of innovation and accessibility of products resulting from innovation. It is a win-win situation for end-users, corporates and society, in general, if open-source projects are encouraged.

3.8.5. Issues in adopting open-source software

A major issue in adopting FOSS is the lack of know-how and the lack of the knowledge about the availability of such options. Many open-source projects have evolved over a period of time to become useful to the technical community. For example, Apache Web Server, one of the most popular products, needs substantial technical expertise to set up and run effectively. Linux operating system, one of the earliest success of open-source initiative, still requires the user to be familiar with some commands. This serves as a deterrent to the normal user – partly due to fear of technology and partly due to the unavailability of user-friendly manuals and educational resources. Also, unwillingness to share code and programming know-how among peers act as a barrier in development of open-source projects. Last but not the least, the lack of public policy to encourage open-source projects is partly to blame. In the US, the government has mandated that at least 20% of the government sponsored software pilot projects must be released as open-source (Scott & Rung, 2016). The developing countries will benefit from such laws mandated by their own government.

3.9. Improvement of digitalization in Bangladesh by using FOSS

Using unlicensed software raises not only security risk but also proves costly in terms of maintenance. It is unfortunate that across Asia-Pacific, Bangladesh has had the highest rate of unlicensed software, at 84 percent, followed by Indonesia and Pakistan, which each has rate of unlicensed software at 83 percent (BSA, 2018). Therefore, Bangladesh can use FOSS to reduce the risk of cyber-attack. The following are some benefits.

3.9.1. Reduction in the expenses

A number of governments have made substantial savings in software expenditure by adopting open source. For example, the South African government has reduced licensing fee for proprietary software by requiring its offices to prioritize open-source software. As a corollary benefit, the government could use old hardware, which supports open-source software, obviating a costly hardware upgrade, resulting in cost saving on both software and hardware. A developing country like Bangladesh could profit immensely from adopting similar policies.

3.9.2. Establishment of a strong safety network

Governments are increasingly worried about the severity of consequences from having to use software of unknown origin and undisclosed logic. There are cries internationally to ban certain software from public due to national security issues. Open-source alternatives are open to government employed security professionals to review and revise. Leakage of sensitive information can be thwarted and privacy of citizens' data can be ensured using open-source packages.

3.9.3. Institution of autonomy to redistribute and copy

Having the ability to replicate systems across many branches in the government, some of which could be located off-shore, rendered flexibility to government to control its IT and operational efficiency. Imagine that all foreign consulates of Bangladesh are using the same open-source software distributed by the government as opposed to purchasing heterogeneous mix of software, where each piece of software is subject to the native countries regulations and taxes. Increased operational excellence in government IT divisions would also facilitate easier adoption of technology.

3.9.4. Stimulus to the local software industry

An oft-cited reason why local IT industry lacks growth is the small size of the local market. Government could help the local industry a great deal by adopting open-source model, where skills of local software professionals are utilized to customize and modify open-source software. This would decrease reliance on foreign vendors as well.

3.10. Private-Collective model producing benefit for the society

Existing economic structure of the world are hinged on the premise that society benefits from innovation. It is, therefore, worth exploring how open FOSS development model brings benefits to the society. There are criticisms to the business model of open-source model vis-a-vis private enterprise model, which we highlight in our discussion. The traditional private investment model of proprietary production of software is also contrasted with that of open-source model. In the end, we show that how private investment model and open-source model go hand in hand in modern business climate (Hippel & Krogh, 2003).

3.10.1. Private investment model

In the private investment model, assumption is that enterprises would innovate, provide goods and service to the users at a cost that is affordable and reap rewards that would sustain and bring profit to the enterprise (Demsetz, 1974). The more that enterprises can innovate, the more value is created to the end-consumers. Modern societies built legal infrastructure to reward innovators by way of patents, copyrights and trade-secrets. The laws ensure that innovators get rewards for their inventions (Arrow, 1962; Liebeskind, 1996).

It is assumed that as a consequence of providing such legal protection to the companies, creator individuals, the society will, in general, generate maximum innovation (Audretsch & Feldman 1996; Audretsch & Stephan 1999; Harhoff et al., 2000). While simplistic, this view ignores the opportunity cost of not opening up the innovation to the third-party. An intellectual innovation is not an end in itself. Rather, the innovation can go through revisions, modifications and enhancements if other innovators are free to obtain the right to do so. By granting innovators sole right to the innovation, the law often deprives other innovators from using the innovation for other noble purposes. While private investment model and laws to protect intellectual property rights generate maximum revenue for the innovator, we argue that it comes at a cost to the society in general.

3.10.2. Collective action model

This model is based on the notion of use and innovation of public goods. As an example, a scientific work/research falls under the purview of this model. Scientific works are often based on the previous works of other scientists (Merton, 1973; Aldrich, 1999; Monge et al., 1998; McCaffrey et al., 1995;

Coleman, 1973; Eyerman & Jamison, 1991; Hess, 1998; Melucci, 1999). If the scientist does not open his or her innovation to the public, humankind are deprived of the merits of innovation not least because other scientists fail to make subsequent enhancement to the invention. The collective action model propounds that all scientific works should be freely available (Olson, 1967). In the context of intellectual property rights of software, one might argue that collective action model produces innovation at a quicker rate and in more volume. Hence, software should be free and any innovation related to software must be made publicly available to maximize societal gain.

While collective action model highlights the idealized version of human endeavor, it lacks in associated problem that arises when the enterprise/innovator is not allowed to reap some benefits from innovation – it is well-known that free-riding is encouraged when innovators wait for other innovators to innovate. By not giving sole rights to reap benefits from innovation, many would-be innovators would simply not have enough motivation to innovate (Olson, 1967). One could argue that similar to the credit/reputation attribution in the education/research domain, society could reward innovators in the industrial domain by providing honor system or even by subsidizing the research. However, this is not practical in the modern society as software innovation is still aligned to its industrial base. Society often does not subsidize innovation sufficiently for the innovators to pursue the innovation solely for the purpose of public good. (Merton, 1973; Stephan, 1996).

3.10.3. Private-Collective model

In the open-source arena, we observe a unique model that is a mix of private-investment and collective action model (Homscheid et al., 2015). In private-collective model, participants in FOSS derive their individual benefits through contributing. The companies benefit by being sponsors of major open-source projects and being part of the development community. Being part of the open-source software both the individual contributor and companies exert some control over the development direction of the project. Some companies stand to gain from selling complementary products or services related to the open-source software. Figure 7 below summarizes the increased value imparted by the private-collective model described in this section.

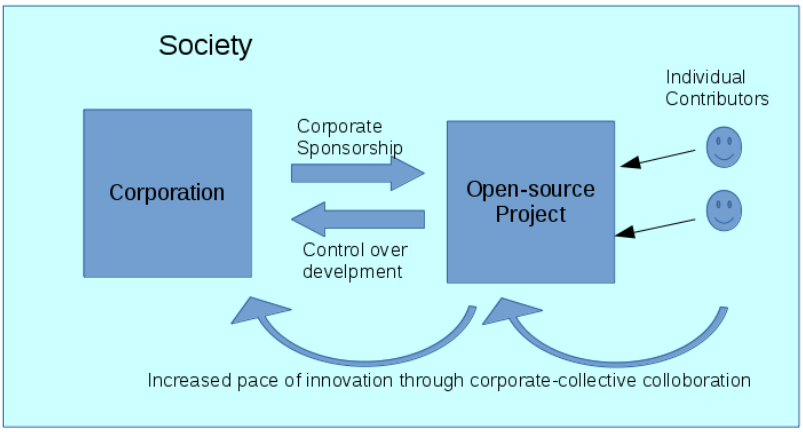

Figure 7: Corporate-collective collaboration open-source model

4. Conclusions

In this paper, we have explored the legal framework that exists in the modern world to protect intellectual property rights related to software and, especially, open-source software. We demonstrated that copyright laws have been expanded to include digital copyright including that of software. To answer our research question as to why adopting open-source licensing could benefit a developing nation like Bangladesh, we have explored the benefits from end-user, individual developer, and corporate perspectives. Our findings demonstrate that both from safety and cost of ownership points of view, adopting open-source is a viable alternative, if not preferable. To answer how society could be benefited from open-source software, we have highlighted that open-source software development increases the pace of innovation, results in more sharing of knowledge and ultimately produces more value to the society. We have listed the potential barriers to adopting open-source licensing in the context of Bangladesh. We brought attention to the unavailability of public policy regarding the fostering of open-source products. During the study, we found that there is a clear lack of study on the subject matter in Bangladesh. We believe that our study opens the pathway to more illuminating studies in the future. Such studies are crucially needed to pave way to the future so that both creators and end users could derive more benefits from the creative work of programmers.

References

Arrow, K. J. (1971). The economic implications of learning by doing. In *Readings in the Theory of Growth* (pp. 131-149). Palgrave Macmillan, London.

Audretsch, D. B., & Feldman, M. P. (1996). R&D spillovers and the geography of innovation and production. *The American Economic Review*, *86*(3), 630-640.

Audretsch, D. B., & Stephan, P. E. (1999). Knowledge spillovers in biotechnology: sources and incentives. *Journal of Evolutionary Economics*, *9*(1), 97-107.

BSA (2018). Global Software Survey- June 2018. [Online] Available at: https://gss.bsa.org/wp-content/uploads/2018/06/2018_BSA_GSS_Report_A4_en.pdf

Campbell-Kelly, M., & Garcia-Swartz, D. D. (2009). Pragmatism, not ideology: Historical perspectives on IBM's adoption of open-source software. *Information Economics and Policy*, *21*(3), 229-244.

Demsetz, H. (1974). Toward a theory of property rights. In *Classic papers in natural resource economics* (pp. 163-177). Palgrave Macmillan, London.

Deshpande, A., & Riehle, D. (2008). The total growth of open source. *IFIP International Federation for Information Processing*. https://doi.org/10.1007/978-0-387-09684-1_16

Directive 91/250/EEC (1991). Council Directive on the legal protection of computer programs. [Online]. Available at: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:31991L0250

European Parliament (2013). Compilation of briefing notes on Legal aspects of free and open source software, Workshop, 9 July 2013. Available at: http://www.europarl.europa.eu/document/activities/cont/201307/20130708ATT69346/20130708ATT69346EN.pdf

Guadamuz, A., & Rens, A. (2013). Comparative Analysis of copyright assignment and licence formalities for Open Source Contributor Agreements. *SCRIPTed*, *10*, 207.

Hars, A., & Ou, S. (2002). Working for free? Motivations for participating in open-source projects. *International Journal of Electronic Commerce*, *6*(3), 25-39.

Hippel, E. von, & Krogh, G. von. (2003). Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. *Organization Science, 14*(2), 209-223.

Homscheid, D., Kunegis, J., & Schaarschmidt, M. (2015). Private-collective innovation and open source software: Longitudinal insights from Linux kernel development. In *Conference on e-Business, e-Services and e-Society* (pp. 299-313). Springer, Cham.

Lerner, J., & Tirole, J. (2003). Some Simple Economics of Open Source. *The Journal of Industrial Economics, 50*(2), 197-234.

Scott, T., & Rung, A. E. (2016). Federal source code policy: Achieving efficiency, transparency, and innovation through reusable and open source software. *Office of Mgmt. & Budget, Exec. Office of the President Memorandum*. [Online]. Available at: https://www.whitehouse.gov/sites/whitehouse.gov/files/omb/memoranda/2016/m_16_21.pdf

Mathan, A. M. P. (2014). Software Protection: International instruments and trends. Available at: http://www.academia.edu/8316248/Software_Protection_International_Instruments_and_Trends

Maxwell, E. (2006). Open Standards, Open Source, and Open Innovation: Harnessing the Benefits of Openness. *Innovations: Technology, Governance, Globalization, 1*(3), 119-176.

Merton, R. C. (1973). Theory of rational option pricing. *The Bell Journal of Economics and Management Science*, 141-183.

Mirdha, A., Jain, A., & Shah, K. (2015). Comparative analysis of open source content management systems. *2014 IEEE International Conference on Computational Intelligence and Computing Research, IEEE ICCIC 2014*, (pp. 1-4).

Friedman, T. L. (2006). *The world is flat: A brief history of the twenty-first century*. Macmillan.

Mozilla (2020). Client Bug Bounty Program. Available at: https://www.mozilla.org/en-US/security/client-bug-bounty

Olson, M. (1967). The Logic of Collective Action. Harvard University, *Press, Cambridge,* MA.

Oracle America, Inc. v. Google, Inc., No. 17-1118 (Fed. Cir. 2018)

OSI (Open Source Initiative) (2020). Licenses & Standards. Available at: https://opensource.org/licenses

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1986). *Numerical Recipes: The Art of Scientific Computation*, Cambridge and New York, Cambridge University Press, 1986, 839.

Rajiv, B. G. (2018). Outline of Outline of Rajiv B. Gokhale in the matter of *CoKinetic Systems, Corp. v. Panasonic Avionics Corporation (2017)* (1:17-cv-01527) District Court, S.D. New York

Scacchi, W. (1989). The USC system factory project. *ACM SIGSOFT Software Engineering Notes,* 14(1), 61-82.

Scacchi, W. (2007). Free/open source software development. 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, *ESEC/FSE* 2007. pp. 459-468.

Sudha (2020). Software protection: International instruments and trends. Available at: http://www.legalserviceindia.com/legal/article-3-software-protection-international-instruments-and-trends.html

Weber, S. (2004). *The success of open source*. Harvard University Press.

WIPO (2016). WIPO Copyright Treaty, 1996. Available at: https://www.wipo.int/treaties/en/text.jsp?file_id=295166#P58_5860

This page intentionally left blank