Chapter 2

The Philosophy of Open Source

'Open source' simply means that the source code of a computer program is openly available for use without many restrictions. This terminology is commonly used to signify a production model for software development. As a production methodology, open source is characterized by numerous volunteers contributing to the process of software development in a systematic manner and the resultant software's source code being kept open. Most of the open source software are free of cost, that way easily accessible and even modifiable to suit personal requirements. The ‘accessibility’ factor makes the open source software highly popular. This in turn tempts the creative users to contribute towards its improvement which make the software technologically superior than its competitors. In the capitalist mode of production, the creative consumers used to play only a passive role. Open source brought in a significant change in the production process by involving creative users as producers of information resources. This resulted in the spreading of a culture of working together to resolve the common challenges. The popularity of this production model and its success in software industry has inspired its adoption into many other fields of technology. It is also being promoted as capable of making the industrial research more efficient to overcome the challenge of access, that current model of production has created. Industries have started pursuing open source as a workable solution to more efficiently organize research and bring down the development cost by utilizing social labor.

Even though the formalization of the concept of open source happened only in the 1990s, the collaborative software development existed from the very early days of computing. In fact, the nearly 70 years history of computing technology has several instances of collaborative research undertaken at different levels to resolve the common challenges faced by programmers. As a unique production model, open
source is often pointed out as an alternative to intellectual property rights. At the same time 'open source' has often been criticized as lacking a solid theoretical foundation making it purely contextual and inapplicable for areas other than software development. Interestingly, open source works within the intellectual property framework itself and uses its unusual licensing terms to reduce proprietary claims by relinquishing several rights to the users. But there exist significant differences in approach even inside the ‘free software’ and ‘open source’ movement itself. In essence its ideological foundation has two distinct views. Supporters of ‘free software’ perceive it as a social movement\(^1\) while proponents of ‘open source’ describe it simply as a convenient practice. Needless to say, there is a lack of clarity as to what exactly is free and open source. In order to fully understand the concept of openness, it is very essential to know the context of origin of open collaborative programming. That will probably give a better insight into the theoretical foundation of open source.

### 2.1 Context of an early cooperation

If open source's focus is on the social organization of software production, its origin is deeply rooted in the constraints on independency in computing. Early generations of computers were primordial and highly complex to handle\(^2\). The capabilities that modern day computers possess like huge memories, fast processors, reliable and vast storage media and most importantly, connectivity were hardly imaginable in the early days\(^3\). It was indeed a very complicated task to make the machine perform a desired task. In the modern computing, software written in popular programming languages take on this job. But during the early years of computing, this was a strenuous task for the programmers as they had to program

\(^1\) Richard Stallman, *Free software, free society: Selected essays of Richard M. Stallman* (Free Software Foundation Inc 2002) 57
without many of the modern day tools. Business houses which owned computers had to substantially invest for developing program, suiting their requirements. This often resulted in reinventing the wheel, as competing companies independently developed software which were intended for identical purposes. Things would have been much better if they had a compiler which could translate higher-level programming languages into machine language. The programmers facing this challenge recognized the obvious step of getting everyone together who were using the machine, regardless of what company they work for and build a compiler that everyone could use\(^4\). Such collaboration across corporate boundaries was a necessity at that time. It was the result of different companies recognizing the commonality of their computing problems and the possible benefits from an inter-company collaboration. This lead to the formation of PACT\(^5\), one of the first cooperative efforts in computing history. Success of this project demonstrated the viability of cross-corporate collaborations in programming. As Paul Armer puts it, 'the spirit of cooperation between member organizations and their representatives during the formation of PACT-I has been one of the most valuable resources to come out from the project'\(^6\). The spirit of cooperation inspired all its participants.

---


5 Project For the Advancement of Coding Techniques (PACT) began in November 1954. In this project North American Aviation, Douglas Aircraft Company, IBM, Ramo-Woolridge, and the RAND Corporation collaborated to develop an automatic coding system for IBM 701. This was a common objective for airliners who were main business rivals at that time. The collaboration ultimately resulted in a working compiler that went through two different versions, PACT-I and PACT-II. Adoption of this standard compiler by all the contributing partners removed the technical barrier for further collaborations on software projects. PACT also demonstrated the feasibility of a cooperative coding project and affirmed the value of cooperation, transforming the culture of business computing and creating a climate conducive to further collaboration. See Paul Armer, ‘SHARE-A Eulogy to Cooperative Effort’ (1980) 2 IEEE Ann Hist Comp 122.

This idea of cooperation was taken further with the introduction of IBM 704\(^7\). This machine was more advanced, powerful and complex to handle than its predecessors. All the programs which worked on early generation of computers were incompatible with IBM 704. So the business organizations interested in installation of this machine had to develop new software which are compatible with this new machine. The success of cooperative computing in the PACT, inspired various organizations to explore the possibilities of a joint effort in connection with the program development for IBM 704. This lead to the formation of SHARE\(^8\). As a collaborative movement SHARE grew rapidly and in a short span of time was able to develop an impressive library of routines which each member could use. Some three hundred programs were developed as part of this collaboration. Without a cooperative effort, each contributor would have faced the burden of developing it alone\(^9\). From the industry point of view, SHARE was indeed a blessing for IBM as it increased the acceptance of IBM’s equipment and helped the sales of 704 machine\(^10\). A notable result of these collaborations was standardization slowly happening in computer programs. This eased cooperation for further innovations.

In 1964, MIT\(^11\) joined into a cooperative endeavor with Bell Labs and

\(^7\) The IBM 704, the first mass-produced computer with floating point arithmetic hardware was introduced by IBM in 1954. IBM 704 was significantly improved over IBM 701 in terms of architecture as well as implementations which were not compatible with its predecessors. IBM stated that the device is capable of executing up to 40000 instructions per second which was a significant improvement in computing power at that time. Introduction of IBM 704 required the users to develop new programs as operating softwares of IBM 701 were mostly incompatible with the new installation. Available at <http://www.columbia.edu/cu/computinghistory/704.html> (accessed 3 Aug 2012)

\(^8\) SHARE began under the initiative of RAND corporation, Lockhead Aircraft Corporation and North American Aviation Inc. Its membership soon grown to more than sixty owing to two factors, one the popularity of IBM 704 and secondly because of the increasing acceptance for a collaborative movement in software production. See Paul Armer, ‘SHARE-A Eulogy to Cooperative Effort’ (1980) 2 IEEE Ann Hist Comp 122, 125.


\(^10\) Paul E Ceruzzi, A history of modern computing (MIT Press 2003) 88

\(^11\) Massachusetts Institute of Technology, played an important role in the evolution of collaborative production in computer programming. See Paul E Ceruzzi, A history of modern computing (MIT Press 2003)
General Electric for developing a second generation time sharing system called Multics\textsuperscript{12}. The project was a failure mainly because the three project collaborators had different goals for the system\textsuperscript{13}. Soon the project was discontinued. But it cannot be denied that it was inspirations from the Multics project that contributed to the development of Unix which is regarded as a significant step in computing history. The Multics project also marked the beginning of MIT becoming actively involved in the collaborative efforts for the advancement of computing technology. This is very significant in the history of modern computing. Open source historians view these early cooperative efforts as significant steps that made the world realize the strengths of collaborative programming and laid the strong foundation for open culture.

Programmers have by this time realized the potential of collaborative efforts in computing. This resulted in a perception that cooperation is better than competition in at least certain realms of computing technology. The spirit of cooperation was significantly influencing the programmers. This period was also witnessing the increasing acceptability of computers in business. Computing was soon identified as a technology with much potential and capable of transforming the then existing way of doing business by increasing efficiency. This encouraged more and more industries to adopt computational tools and achieve organizational excellence. But at the same time, computing technology was growing at a rapid pace and this required the industry to undergo significant changes whenever new generation of machines are introduced. This cemented the increasing realization that the culture of sharing is a necessity in the context of emerging technology.

\textsuperscript{12} Multics- 'Multiplexed information and Computing Service' was an influential early time sharing operating system. The project started in 1964 in Massachusetts. Originally it was a cooperative project led by Massachusetts Institute of Technology, General Electric and Bell Labs. But Bell labs pulled out from the project in 1969.

\textsuperscript{13} Steve Weber, The success of open source (Harvard University Press 2004) 24
2.2 Legal Foundation of Open Source

The legal foundation for open source could be traced back to the popularity of Unix as a technologically advanced operating system and the legal inability of Bell Telephone laboratories to sell it. The conceptual origin of Unix happened in the Multics project jointly undertaken by MIT, Bell labs and General Electric. After the failure of Multics, researchers at BTL decided to take it forward on a much smaller scale. This resulted in the development of Unix. Unix followed a modular design which provide a set of simple tools that perform a limited well defined functions. Unix was also the first portable operating system that can work across numerous platforms. This immediately made Unix popular among the programming community. By 1974, BTL was getting requests for copies of the Unix operating system. This looked like an excellent business opportunity for AT&T which is the parent company of Bell Labs. But AT&T was unable to utilize this opportunity because of a legal impediment under the Antitrust law that they encountered earlier. On January 24, 1956, Western Electric and AT &T had entered into a consent decree with Eisenhower Administration Justice Department for the settlement of a Sherman Antitrust Act complaint. The consent decree specified that AT &T and Western Electric should not engage in manufacturing or sales outside of telephone, telegraph and “common carrier communication services”. The legal interpretation of this consent decree made it clear that AT & T cannot pursue any business other than phones and telegrams. Unix was a software, not a telephone or telegraph, which meant AT &T could not sell it. AT &T, Western Electric and Bell Telephone Laboratories were also required to license their patents at nominal fees. This signaled the governmental intention to accelerate the dissemination of communications technology and other innovations that were happening at Bell labs. This also gave software, a legal and contextual interpretation as a pre-

competitive tool which would play a critical role in the future of collaborative efforts in software development\textsuperscript{16}.

Unix easily became a widely sought after operating system due to its many features. At that time, it was the only software available which could run on the less expensive mini computers owned by University departments. Due to its inability to sell Unix as a commercial product, AT&T started licensing it away to academic institutions for a nominal fee of 150 $\textsuperscript{17}. From the very beginning, AT&T made available the full source code of Unix under a trade secret agreement. The complete package of Unix, included a tape of source code in C language and its manuals. The AT&T’s license for Unix included a right for the programmers to alter the source code and share changes with other licensees\textsuperscript{18}. This allowed the programmers to experiment with the source code and adapt it to suit their own requirements. Thus the programmers were allowed to modify and improve the source code at their will and share changes with other licensees. This has to be seen as the legal foundation of open source which gave away the programmers a creative freedom of new dimension in the form of a legal right.

A new collaborative culture was evolving which took the already existing cooperation into a new level. The flexibility of Unix made it a highly desired operating system. Programmers were free to alter Unix to suit their diverse needs and then to share their improvements with likeminded users. AT&T was not providing any support for Unix like bug fixing. This necessitated a collaborative network among the programmers to support each other in bug fixing. It also became a clear incentive for Unix programmers to share support and bug fixes with one another. Very soon a culture of sharing improvements spread among the

\textsuperscript{17} J. Naughton, \textit{A Brief History of the Future–The origins of Internet} (Phoenix 2000) 176.
programmers of Unix and collaborative effort began to be seen as beneficial to all programmers alike. Unlike today, there wasn't any formalized organizational platform for the working of these collaborations. In those days, Unix spread mainly through physical means like tapes, disks etc. Programmers and research fellows who were traveling back and forth between various institutions used to carry these tapes and disks. Within a year, Unix could create an active set of programmers even in England, Australia and Japan. By 1975, there were at least forty institutions using Unix in United States. Programmers in each country organized meetings and informal gatherings to share their ideas, support and bug fixes. This soon strengthened the community support for Unix. This could be seen as the evolution of an early form of present day open source collaborations. History evidences that there existed a physical community of Unix programmers, long before anything like a virtual community could have been imagined. This resulted in the ‘culture of sharing’ being accepted as a standard norm in the programming communities.

2.3 The programming culture of Open Source

The origin of open source’s programming culture could be traced back to the hacker communities of MIT. In the 1960's MIT's Artificial Intelligence Lab became a major center for research on communication technology and computing. The ideals of openness, sharing and collaboration which was integral to the academic culture at MIT had an important role in the evolution of Hacker's philosophy. In academia, access to the base materials was seen as a prerequisite for

---

21 The Tech Model Railroad Club (TMRC) is a student organization at the Massachusetts Institute of Technology (MIT), and is one of the most celebrated model railroad clubs in the world. This organization has a historic role as a breeding ground of hacker culture. See Steven Levy, *Hackers - Heroes of the Computer Revolution* (Boston Doubleday Dell Publishing Group Inc 1984).
continued creativity\textsuperscript{23}. Access to the information is a must to improve existing things. If information required for improvising existing things is enclosed, creativity is strangulated. A free exchange of information particularly when the information is in the form of a computer program allows greater over-all creativity\textsuperscript{24}. In the early days of computing there wasn't a clear distinction between hardware and software. So was the difference between a programmer and a user. The user-programmers of those days were required to make the machine do the intended task. Programmers of those days were mainly electrical engineers. Unlike today, programming was done mainly in machine level languages. The advent of collaborations eased the job of programmers in many respects. When cooperation became a necessity due to the constraints of new technology, it was the engineers who were put to work in the new atmosphere of sharing and collaborating. It eliminated the redundant effort programmers had to spend developing the program independently. Cooperative effort was indeed a new experience for the programmers considering the industrial work culture of that time. They soon realized the possibilities of social cooperation in computing and the potentials of collaborative creativity\textsuperscript{25}. Cross-corporate collaborations developed a pre-competitive atmosphere of sharing innovations for accelerating the advancement of technology. Sharing of source code created a new culture where programmers never hide the base materials of their innovation. This allowed programmers to learn from others and resulted in continuous improvisation of technology. This became mutually beneficial for the programmers and it often led to mutual admiration of each other's programming abilities. The peer-recognition received during such collaborative programming became a significant incentive for creativity. The programmers started to enjoy the freedom they got to experiment with the code and they effectively utilized this opportunity to improvise their skills. Programmers' experiences from this collaborative culture soon influenced their

\textsuperscript{25} Paul Armer, `SHARE-A Eulogy to Cooperative Effort’ (1980) 2 IEEE Ann Hist Comp 122, 129.
thinking and way of approach to computers. By the 1960's when computer moved into mainstream business operations, demand for programmers grew rapidly. Programmers of those days perceived themselves as innovators or crafts people. They saw their culture as one of artisanship more than engineering. They started believing that it is possible to create art and beauty on a computer. They found programming as an art. It was definitely the artistic impulse in programmers that enabled them to play with algorithm and come up with its unexpected combinations.

University-Industry collaborations made it possible for corporate programmers and academicians to work together on same projects. Business considerations were no longer a priority when the research continued in an academic environment. Programmers at Berkeley and MIT regarded source code as a base material for programming. For them access to source code permitted flexibility and adaptability of software to suit their needs. They cherished this 'openness' and started the culture of sharing source code with anybody making a request. Source code soon became recognized as a tool for creativity. Software sharing and co-creation soon became a culture rather than a necessity. This lessened the gap among programmers that was widening in the course of time. A practice which started purely on the basis of convenience was slowly getting engraved into the heart of programmers. User-programmers always responded with improvisations which were incorporated into the basic version and distributed again. The collaborative culture was evolving to a new level. Lack of accepted standards was something which stood in the way of collaborative creation. But the programmers were skeptical that premature standardization would probably advantage different people and

---


companies. Programmers believed that creative freedom must be given importance than the benefits of standardization. Standardization would have unidirectionally diverted technological growth. It would have constrained creative freedom. Creative freedom was soon becoming the spirit of open software community. This along with certain distinct notions of academic life significantly influenced a culture that was evolving among the programmers. Steven Levy asserts that if everyone could interact with computers with the same innocent, productive, creative impulse that Hackers did, their ethic might spread through society like a benevolent ripple and computers would indeed change the world for the better. This culture indeed played a significant role in formalization of current open source model.

2.4 Formalization of Open Source Model

A cooperative model of software development was already in place by the late 1970s. The model was characterized by the original innovation becoming substantially improved by user-programmers through their contributions. It was happening in a grossly unorganized manner. The system in place was only a sharing culture, where programmers shared each other’s innovations. This has been very successful in contributing to the development of computing technology. But collaborative production required some form of organization. Hacker culture had resulted in pushing the programmers to the verge of playfulness. They experimented with new software and made numerous improvements. This resulted in multiple versions of same software without clear distinctive marks to recognize them. Distribution process was happening through physical means. This resulted in software being misnamed and spurred confusion. Programmers started to feel that there should be some mechanism to distribute software, receive modifications from user-programmers, update original version with improvements and redistribute it.

29 Steven Levy, Hackers-The heroes of computer revolution (Dell publishing Group Inc 1968) 40.
This was seen as important for meaningful collaboration to take place. Such a system will make it possible to optimally utilize the creative ability. Ultimately this will be beneficial to programmers as they will have better access to resources. But it took a long time in the history of computing for such a system to evolve.

In 1975, Berkeley students\textsuperscript{30} improvised Pascal interpreter and turned it into a highly sought after programming system. The news about this new Pascal system at Berkeley, spread around the Unix community. So everybody wanted a copy of it. Thus Bill Joy at Berkeley started distributing a package of tools and utilities which he called 'Berkeley Software distribution'. User-programmers who received Berkeley Software Distribution responded back with feedbacks and their own improvements. In 1978, a second BSD release\textsuperscript{31} was done incorporating these improvements. Thus BSD soon started improving on its own. By this time, open source culture had become so popular that it was not just computer scientists but even high school students trained in programming had started contributing to the community\textsuperscript{32}. Open source community was growing beyond programmers and started involving the real users who are aware of programming. Their contributions as bug fixes and utility improvements, were shared with the entire user community and were incorporated into future releases of Unix. This finally culminated into the release of Unix version7 which was significantly advanced and got ported into variety of machines. DEC\textsuperscript{33}, a major system manufacturer of that time, used to develop its own proprietary operating systems. In fact, they had never taken Unix seriously even though majority of their customers were buying DEC to run Unix. But when version 7 was released, DEC wanted it to be ported to their machine also.

\textsuperscript{30} Bill Joy and Chuck Haley were the graduate students, who improvised the Pascal interpreter. They did this by working on the Unix version 6 brought to Berkeley by Ken Thompson who was there on Sabbatical from Bell Labs.

\textsuperscript{31} The Second Berkely Software Distribution (2BSD) was released in May 1979.


\textsuperscript{33} DEC- Digital Equipment Corporation was a major American Company in computer industry and a leading vendor of computer systems and peripherals from 1960s to 1990s.
In 1978, DEC approached Bell Labs for developing a Unix version that can be ported into their system. This resulted in the development of a 32V port version of Unix 7. This received wider attention and everybody wanted a copy of it. This made AT &T realize the potential commercial value of it. But even at that time AT&T was bound by the consent decree under Antitrust law. Thus in 1978 AT &T decided to license 32V port design only to Berkeley. The programmers at Berkeley soon reworked the system, rewriting some parts of kernel and porting over additional utilities. The result was a completely new Unix operating system. Thus 3BSD became the first complete operating system release from Berkeley. The BSD Unix was significantly improved through feedbacks and lead to the release of newer versions. At the same time AT &T was also moving towards commercially oriented releases of its Unix products.

The year 1974 saw US Government initiating another Antitrust suit against AT &T, Western Electric and Bell Telephone Labs alleging that they acted as monopolists in a broad variety of telecommunication services. This was finally settled in 1984 which required disinvestment of operating companies from AT &T. This resulted in Bell labs becoming autonomous and the dissolution of Western Electric. Thus AT &T became liberated from the 1956 consent decree and thus free to enter the computer business. AT &T soon created a separate division called Unix system laboratory. In 1988, the licensing fees of Unix skyrocketed to around $100,000\(^{34}\). The BSD Unix, thus became an alternative to its highly priced counterpart. Even at that time, the BSD version of Unix contained significant percentage of code that was developed by Bell Labs. Soon AT&T bought 20 percent of Sun Microsystems and announced mutual plans of moving together in the proprietary direction. This alarmed other major users of Unix. In 1988, Apollo, DEC, HP, IBM, Bull, Nixdorf and Siemens joined hands to create 'Open Software Foundation'. The objective of this organization was to build their own Unix which is free of AT &T licenses. But this plan didn't materialize for many reasons.

---

By that time, AT&T had started attempts to enforce ownership claims over Unix. Even though Unix contained code written inside Bell Labs, it also contained improvisations done by numerous contributors. This included students, scientists and computer enthusiasts. The version of Unix at that time was in fact the result of substantial improvement by many contributors who were collaborating across institutions and corporations. Thus the attempt of AT&T to enforce proprietary claims didn’t go well with the programming communities. It also fueled their skepticism towards big corporations and intellectual property regime. Privatization of Unix also signaled that intellectual property rights could easily be used to rob authors of their work. This significantly influenced the cultural mindset of programmers. Under BSD license terms, it was legal for the user-programmers to develop a proprietary product out of BSD code. Thus the user-programmers had the freedom to redistribute improved versions as proprietary software. In 1992, 'Berkeley Software Design Incorporated' a newly formed company started selling an improved version of Unix for just under $1000. This emerged as a direct threat to AT&T’s highly priced version of Unix. This resulted in AT&T filing a law suit seeking a preliminary injunction to bar distribution of BSDI software on the ground that BSDI had infringed copyright and misappropriated trade secret. AT&T’s law suit claimed that BSDI have stolen their proprietary source code. BSDI defended with the argument that they were only using the freely distributed source code released by Berkeley. The court accepted this contention and the only option left for AT&T was to initiate action against its longtime collaborator, University of California. Even though AT&T re-filed its suit against both BSDI and University of California, the case didn't go for trial. When AT&T sued BSD Unix for intellectual property violation, the only option left for open source articulators was to perform a


clean room operation and rewrite that part of BSD source code which is derived from AT &T Unix. The objective was to rewrite the entire part which was relying on AT &T’s code. Several Unix programmer group meetings were organized and the collaborative effort resulted in rewriting the Unix\textsuperscript{37}. During this exercise it was promised that all contributors would have their name listed in the credits specifying the utility that they wrote\textsuperscript{38}. There evolved a collaborative system where contributors are rewarded by their name being included in the credit. Credits also serve as documentation for the contributions. This formalized the production process of open source. Finally, Berkeley released its new version 4.4.BSD-Lite in 1994. AT &T agreed that they would not have any claim against anyone using this software. This resulted in BSD being freed from all legal impediments of proprietary claims. The present day open source model of software production is characterized by a decentralized and distributed development process accepting voluntary contributions. Even though origin of this culture could be traced back to cooperative movement in 1950's, its real formalization as a production methodology happened only in the 1990's.

2.5 Theoretical foundation for Open Source

The origin of collaborative programming was purely a convenient practice as evident in the history. Initially the communities were only focused on releasing the source code along with the software. But the practice also started significantly influencing the programmers who started professing it as a culture. With the growth of the movement, different perspectives have also emerged about it. Some continued to perceive it as just a convenient practice while some others attached ethical and moral considerations to it. The practical approach is that open source is a development method

\footnotesize{37 Keith Bostic took the initiative and solicited contributors at Unix user group meetings. He is an American computer programmer, who joined Computer Systems Research Group (CSRG) of University of California in 1986.}

for software that harnesses the power of distributed peer review and transparency of process\textsuperscript{39}. They identified better quality, higher reliability, more flexibility, lower cost etc. as the advantages of open source. They believed that open source doesn’t require a theoretical foundation as it is only a practical goal. On the other hand, articulators of software as having moral and ethical dimensions believed in its freedom. This was happening in the late 1980’s, when the software industry was drifting swiftly into proprietary direction and the programmers needed an alternative thinking to grow. The origin of ‘free software movement’ was as an attempt to theorize the practice of collaborative software development. As the whole movement is as a reaction towards the restrictive use of intellectual property by software vendors, proponents of the movement attempted to project it as a direct challenge to the intellectual property based software model. The underlying principle behind this movement was that the use of computers shall not lead to people being prevented from cooperating with each other. Richard Stallman was the main proponent of this approach. He was an inheritor of the hacker culture at MIT\textsuperscript{40}. With the collapse of MIT hacker community, Stallman took the responsibility to rebuild the open source community by laying a theoretical foundation for it. The operating systems used in MIT were the products of collaborative effort from the entire community. It was open for anyone to experiment with it and to do whatever he or she wanted to. Cooperation was seen as a way of life which was completely changed with the coming of proprietary software. The trade secret agreements of proprietary software require the programmers not to share the software with anyone. Stallman saw this as forcing the programmers to betray the society. According to him this should not be the way to deal with technical information and humanity must be allowed to benefit from knowing it. For him the purpose of science and technology is to develop useful information for humanity to help people live their lives better and if the law requires withholding of such information, it is betrayal of humanity.

\textsuperscript{39} See for details <www.opensource.org/history> (accessed 15 September 2012)
\textsuperscript{40} Steve Weber, \textit{The success of open source} (Harvard University Press 2004) 46
Stallman see software not as just a tool to run computers, but as a manifestation of human creativity and expression. He emphasizes on the intellectual input of a programmer and the joy of creation he receives through collaborative effort. Software is seen as a key artifact of a community that existed to solve problems for the common good. For him proprietary software is against the moral sentiments of society. He perceived intellectual property rights as a hurdle in the way of collaborative production. Thus proprietary software is morally bad even if it has any practical goodness. It allows owners to have control over how others use information stored in the form of digital technology. For him, open source is a development methodology while 'free software' is a social movement. Free software movement perceives non free software as a social problem and free software as a solution. Free software contributes to the human knowledge. The focus of free software is 'freedom and cooperation'. Stallman was a strong opponent of intellectual property rights. He articulated for its complete rejection and termed it as ethically wrong. But it is an irony that Stallman had to rely on the same property philosophy to develop his model of free software. This was necessitated because he realized that keeping the software out of intellectual property protection will expose it to forking. Putting the software in public domain enable user-programmers to build on it and redistribute it even under proprietary terms. Stallman found this as a threat for free software movement and decided to rely on intellectual property rights to develop a licensing framework that will prevent forking and at the same time will facilitate its use in future creations. He thus pioneered the idea of utilizing the existing mechanism of copyright law to preserve access for all to the software developed by free software communities. The innovative idea culminated into a copyright based licensing system which uses property rights to preclude forking and at the same time ensure continued open access to the program. He thus drafted the ‘General Public

---

Chapter 2  The Philosophy of Open Source

License which guaranteed the end user the freedom to run, study, share and modify the software. Thus not only the user-programmers but also ordinary users of the programme were given the freedom to share the programme with other persons. Thus the movement of free software moved beyond the programmers and reached the common public and hence treated as a social movement based on the principles of freedom to use. The first version of GPL was released in February 1989. This was subsequently modified in 1991 and 2007. The theoretical reasoning of the term 'free software' thus represent four freedoms which are numbered from zero to three. They are (0) Freedom to run the software for any purpose, (1) freedom to modify the program to suit ones needs, (2) freedom to redistribute the original version and (3) freedom to distribute modified version. The use of the term 'free' is not with reference to price, but with reference to these freedoms. Stallman himself have pointed out that the use of the term 'free' may be confusing. Stallman’s conception of freedom is limited to the rights that he describes in GPL. It is no way close to the concepts of ‘public domain’ or even ‘social ownership’. His definition of freedom is so drafted to ensure access to code and its unhindered use so long as the user complies with the license terms. A person redistributing original version or distributing modified version of software can do it either gratis or for a fee. So free software may not always be available free of cost.

For the proponents of free software, if a user is not allowed to run the software in the way he likes, it is a restrictive program. It limits ability of a user to work with the program. Freedom zero allows the programmer to run the program for any purpose in the way he likes. If the programmer doesn’t have this freedom it impairs his ability to fully utilize the program. Proprietary vendors often restrict the user-programmers and ordinary users of software except for some specified purposes. Stallman sees this as affecting people's morale and making their life miserable. The freedom zero is specifically designed to overcome this challenge.

42 Available at <https://www.gnu.org/licenses/old-licenses/gpl-1.0.html> (accessed 28 December 2012)
Freedom one allows the user to change the program to suit his needs. This enables the user-programmer to utilize his creative abilities to improve the original version. Freedom two is the freedom to help fellow beings by distributing copies of the program. Sharing of useful knowledge has to be seen as fundamental to the social life of human beings. The purpose of society itself is to teach the spirit of cooperation. It is a natural instinct and part of human nature. Proprietary software restricts the user’s ability to share useful knowledge and thus make it impossible to collaborate with his fellow beings\(^\text{43}\). Freedom three thus allows the user-programmer to create a social life for himself by publishing an improved version of the software that can be freely used by even ordinary users. This is very important both practically and psycho socially. If this freedom is not available, the concept of community cooperation will not happen. It will detrimentally affect the spirit of scientific cooperation. Scientific cooperation is a must for scientific advancement. Stallman’s theoretical arguments and his quest for freedom need to be understood in context of hacker culture of which he was a part. The hackers of 1970s were significantly influenced by the spirit of cooperation. More than anything they were very much attached to the machine that everything else was secondary. Stallman’s argument resonates hacker ethics of those days. But hackerdom represented lack of restrictions especially of proprietary nature. Even though Stallman draws the foundation for many of his arguments from hacker culture, his reliance on property to build a model of collaborative research does not fit well in it. He intentionally separates his stand from the use of a much general term 'open source' by using 'free software' to explain his ideals. His attempt is to defend creative freedom against proprietary software on the grounds of morality and ethics. He perceives that proprietary software which forbids cooperation and compels him to betray others as morally wrong. The whole focus of forming the free software movement was in reality to resist the use of intellectual property to create enclosures that hinder the

use of code. But it is very interesting that the movement had to rely on the same property rights to sustain its creations in an open framework. Even though Stallman was successful in giving the free software movement a new lease of life at a time when the whole software production was moving in proprietary direction, its charm of attaching moral righteousness to programming didn't last long. Many of the programmers found the philosophical overload in free software as impractical and unwanted. The viral clauses of the GPL also clashed with the pragmatic view of many programmers\textsuperscript{44}. It also negatively affected the ability of programmers to use proprietary code even if it is technologically superior. The GPL was inflexible in its approach towards proprietary code and this was seen as a major impediment for commercially utilizing free software. This inflexibility was viewed by many as a different form of enclosure that restricts the ability of user-programmers to use the code in the way they like.

As opposed to Stallman’s articulation for ‘free software’, open source movement emerged in the late 1990’s to liberate the movement from the ideological and confrontational connotations of the term ‘free software’. The Open Source Initiative\textsuperscript{45} started by Bruce Perens and Eric S Raymond in February 1998 played an instrumental role in formalization of this sect as they adopted a more practical definition\textsuperscript{46} for open source which is devoid of any philosophical overtones. Freedom to redistribute, open access to source code, freedom to create derivative works and distribute them, freedom to bundle with proprietary software etc. are some of the features of open source definition.

Many people view open source as a wider concept and free software as a subset of it. While the focus of free software is on the freedoms, open source is

\textsuperscript{44} See for full text of GPL \(<https://www.gnu.org/licenses/gpl-3.0.en.html>\) (accessed 28 December 2012)

\textsuperscript{45} See for details on open source definition \(<https://opensource.org/osd>\) (accessed 28 December 2012)

\textsuperscript{46} See \(<https://opensource.org/docs/osd>\) (accessed 28 December 2012)
simply a model for collaborative production aimed at technological advancement and its economic exploitation. While the free software licenses stipulate that any redistribution of modified version shall also be in accordance with same license terms, open source does not have such restrictions. Proponents of open source considerably differ from that of free software in their approach towards proprietary software. They are willing to co-exist with the makers of proprietary software and feels that the issue of whether a software is open source or not is only a matter of practicality. Open source movement also relies on the same copyright based licensing practice which was followed by free software movement. But open source movement is more focused on the practical benefits of being open, while free software movement is more emphasizing on moral rightness and the importance of creative freedom. Many perceived free software licenses like GPL as complicated and restrictive. This is because GPL requires any derivative works also to be distributed under same GPL license terms. While open source licenses like BSD license requires only acknowledgement of original authors and poses no restrictions on usage of source code in derivative works. These kinds of permissive licenses permit the users to even incorporate the open source code in a proprietary product. Lesser restrictions enable software developers to build on open code and redistribute the modified code under a different license. This flexibility of open source approach soon became popular among the programmers. While the free software failed in becoming main stream, open source soon emerged as a major cultural and economic phenomenon. Its openness to commercial exploitation attracted even the industry.


48 See General Public License GPLv3 - Clause 5 which reads ‘Conveying modified source versions - c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it’.

49 BSD license is available at <https://www.debian.org/misc/bsd.license> (accessed 28 December 2012)
This resulted in different groups, industries and organizations taking up the open source development model and drafting their own versions of open source licenses based on their priorities. These licenses depict considerable variations in their approach with regard to linking with code published under a different license, distribution of code to third parties, approach to patent, sub-licensing etc. While the free software licenses like GPL are complex and lengthy, open source licenses are mostly simple and minimal in content. Apart from these conceptual variations, some companies may also choose to have their own license to project themselves as open source champions and to market their social contributions. All these have in fact lead to license proliferation characterized by existence of numerous licenses which has many similarities and dissimilarities. There are around two million open source projects going on around the world and they use around 1400 unique open source licenses\(^5\). This has emerged as a new challenge on how to comprehend these license terms and its amenability for commercial uses. Open source Initiative have adopted a system for certifying open source licenses based on their conformity with open source definition\(^5\)1. To confirm with open source definition, a license, will have to facilitate redistribution without any fee, must allow distribution of source code, should allow modifications or derivative works and must not place restrictions on any other software that is distributed along with open source software. So far OSI have certified around 83 categories of open source licenses\(^5\)2. As of now, MIT\(^5\)3 license is the most used open source license in the world\(^5\)4.

Open source’s less restrictive approach or rather the permissive approach has

---


51 See for details <https://opensource.org/osd> (accessed 28 December 2012)

52 See for list of licenses certified by Open Source Initiative <https://opensource.org/licenses/alphabetical> (accessed 28 December 2012)

53 MIT license is a permissive open source license. It can be accessed at <https://opensource.org/licenses/MIT> (accessed 28 December 2012)

54 See for rankings of most used open source licenses <https://www.blackducksoftware.com/top-open-source-licenses> (accessed 16 November 2017)
been successful in making it more popular than the free software. While ‘free software’ is often being tagged as too restrictive, many industries started embracing open source by the late 2000s as a viable model for software development. Microsoft, IBM, Oracle, Google etc. were no exceptions to this trend. Social media sites like Facebook, twitter etc. are all powered by open source software. Industry, government institutions and other organizations perceive open source as an opportunity to reduce development cost, increase the development speed and make the system more secure. It is interesting to note that 9 out of 10 super computers in the world now run on Linux. Recent surveys show that more than 70% of the companies around the world use open source software in one way or another. Of them around 65% is contributing to the open source projects as well. This evidences the increasing acceptance that open source is earning in the corporate world. The dependence on open source software is going to be higher in the coming years especially due to the big data revolution and the requirement for analytics tools. The social benefit resulted from this movement for ordinary users of software is the availability of different kinds of programmes at affordable cost.

It is interesting that the mainstream discourse on the theoretical foundation of open source is more centered on the openness of the code and completely discards its relevance in developing a new perspective to theories of human motivation and creativity. More than its practical usefulness in facilitating collaborative production, the working of open source lead us to a fascinating puzzle as to whether it require any incentive for a person to be creative. Though the question appears to be a very simple one, it challenges the fundamental foundations of intellectual property jurisprudence. For long, theorists have justified the intellectual property rights by

---


relying on the incentive theory. The open source phenomenon could be looked at as a classic instance of how creativity can flourish even without any economic incentives. Research literatures on the notion of ‘being creative’, traces out its history to the Greek culture\(^57\). The concept of genius in Greece was associated with mystical powers of protection and good fortune. It represented the individual’s ability to both construct and destruct. However, it is far different from the modern concept, as art was seen as mere imitations. In the Roman view, creativity was generally seen as a male capacity the only exception of which was giving birth. In the Judeo-Christian traditions, creativity was the sole province of God and it was seen as a matter of divine inspiration. Due to the influence of Christian theology the view that all creativity emanates from God and thus have divine origin, dominated the western culture. This is primarily because as per the Biblical notion, everything is God’s creation and human beings do not have the potential to create something new. In Europe, all creative activities of medieval era were church related such as paintings, sculptures, architecture, libraries etc. In the western medieval art, almost all of the themes were derived from Bible. It was only during renaissance the modern concept of creativity as the result of individual genius emerged. This resulted in assertion of a new view point that human beings are responsible for existence of at least certain things. Creativity started to be seen as originating from the human mind as an individual’s ability to bring something new into being. As a result of human success in science and technology, people began to exalt individual rights and gained better ability to understand the universe. This cemented the romantic individualism, where creations are perceived as originating from the abilities of individual. The modern legal construct of intellectual property rights is also based on the assumptions that creativity is individualistic and it can be promoted by extrinsic incentives.

\(^{57}\) Igor N Dubina (et al.), ‘Creativity through a cultural lens : The dichotomy of “The West” and “The East”’ in Igor N Dubina and Elias G Carayannis (eds), *Creativity, Innovation, and Entrepreneurship Across Cultures: Theory and Practices* (Springer 2016) 31
Chapter 2

The Philosophy of Open Source

Generally, people have a tendency to view creativity as a mysterious solo act. But with the growth of modern scientific psychology, it has become clearer that creativity has a social and cognitive dimension as well. Thus creativity has to be seen as a collectively imagined and produced activity. It is a collaborative effort in which both the individual author and the society have a role. Creativity is thus very much a social phenomenon. The socio-cultural heritage creates an inspirational link which connects even artists who lived centuries and continents apart. It is the product of a superlative interaction between 'gifted' individuals and their culture. Thus creativity can never be seen as a momentary impulse. It is brought forth from elements which already exist in the culture and community. Those elements which come together as creative output and the meanings they convey; are all part of the larger culture of which the creator is a part. The authorial role is thus limited to absorbing and interpreting the cultural and social properties of life and then translating it using the creative ability. It is said that, when a writer, musician or theater artist involves in creation of a new work, his imaginative capacity will sequence out his past experiences, his social understandings and prior works of other artists which he enjoyed. Thus his new creation will be the product of this cultural and social experience that he had in his life.

Gordon, in one of his writings, reminds us that all artists receive a gift ‘a tradition and world they have not made’. Authorship is thus the transformation of socio-cultural input gained from the author's experience in his interactions with the society. In a generalized sense, thus all creative endeavors are building upon what is existing. All new works, thus owes much to its predecessors and become a building block for its successors. Social interactions are thus a must to ignite creativity. It would have been impossible to produce creative ideas without social relationships.

Public institutions such as debating societies, libraries, cafes, cabarets and the modern day social networks facilitate the critical comparison and cross-fertilization of ideas. Social interactions are necessary to make an individual realize his talent and then to transform him into an ‘author’. In the process, society enriches his knowledge, sharpens his thought and infuses creative ideas that enable him to translate his talent into authorial creation. The social dimension of creativity thus makes us realize that the exposure to existing works and the liberal freedom to reuse and reshape others’ material is unavoidable in a creative society. The ability to access and consume the products of creativity is thus an essential element of social life. Sustained creativity also demands the freedom to disseminate one’s creativity by making it available to his fellow beings. In such a context, human beings’ innate urge to be creative will alone be sufficient to ensure creative production and thus the requirement of any extrinsic incentive will become irrelevant. In reality the attempt through ‘open source’ is to create such a culture where there are lesser restrictions to create and disseminate information. Open source, as everyoneknow work on voluntary labour of individuals who work together as a community. Research have pointed out that inside open source communities, members feel an intrinsic connection with other members and a collective sense of separation from non-members. This is very much similar to the social life that human beings enjoy in their real life. It allow individuals a greater opportunity to express their creativity, enjoy their work and experience a sense of satisfaction and accomplishment. Inside the community, the participants are free to access each other’s creations. In open source, intrinsic motivation have significant importance as contributors have a high

---


degree of autonomy and self-determination and are valued for their competence. Open source offers an opportunity for readily satisfying human needs for individual autonomy and competence. Some theorists argue that software development is inherently motivating task as it is complicated and creative. This is based on the assumption that there is a human thirst to be creative. In the process of sharing creativity to the fellow beings, a person builds his reputation and thus earns a higher status within the community. At the same time, he will be benefited by the additions and modifications that others make to his creation. The society will also benefit from the open source creation as the product will be open for all to use and enjoy.

The social relevance of open source is in creating an atmosphere which is more conducive to creativity. In the process, it breaks away the barriers of access and facilitates collaborative creation. The social benefit that the society gains is the ability to use the creation which is subjected to constant improvement by the community. The open source process will also contribute to the enrichment of community knowledge. Open source is in essence a form of social production where free labour is organized to collaborate and contribute towards programming. The open access model is so constructed to facilitate creative process without hurdles. Open source must therefore be understood as the resultant of a greater realization about the social dimensions of creativity. As a production model it successfully exploits the social dimensions of creativity. Thus the real theoretical foundation of open source is its exposition as a production model based on free labour which takes into account the social dimensions of creative production. It is this realization that had led to its wider acceptance as an efficient framework to organize collaborative production.

---


2.6. Conclusion

When the seeds of ‘open source’ were sown as early as in 1950’s, it was just a convenient practice for better computing. Constraints of an emerging technology necessitated cooperation beyond all considerations. This was facilitated through social sharing of ideas. The formalization of the collaborative production process increased its efficiency. The attachment of moral and ethical values to open source has to be seen as concerted effort to magnify the intrinsic motivational factors of participants. The theories of intellectual property, generally assume that extrinsic incentives are required to drive innovative activity. But open source is a strong evidence that creativity has a social dimension as well and the intrinsic motivational factors are sufficient enough to make a person creative. Open source has now emerged as a powerful production model to fully exploit the social dimensions of creativity. The increasing acceptance of open source by business organizations signifies its emergence as a viable business model. In this context, the historical positioning of open source as an alternative to intellectual property led capitalist mode of production seems to be overblown. This necessitates a deeper examination into the open source model to understand the property relationships it create and critically examine whether it is a true alternative for the intellectual property model.

---