SUBJECT –OPEN SOURCE TECHNOLOGY

[BCA-IIIrd yr]

Open-source software (**OSS**) is computer software that is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software and its source code to anyone and for any purpose.[1][2] Open-source software may be developed in a collaborative, public manner. Open-source software is a prominent example of open collaboration, meaning any capable user is able to participate online in development, making the number of possible contributors indefinite. The ability to examine the code facilitates public trust in the software.[3]

Open-source software development can bring in diverse perspectives beyond those of a single company. A 2024 estimate of the value of open-source software to firms is $8.8 trillion, as firms would need to spend 3.5 times the amount they currently do without the use of open source software.[4]

Open-source code can be used for studying and allows capable end users to adapt software to their personal needs in a similar way user scripts and custom style sheets allow for web sites, and eventually publish the modification as a fork for users with similar preferences, and directly submit possible improvements as pull requests.

Definitions

The logo of the Open Source Initiative

The Open Source Initiative's (OSI) definition is recognized by several governments internationally[5] as the standard or *de facto* definition. OSI uses *The Open Source Definition* to determine whether it considers a software license open source. The definition was based on the Debian Free Software Guidelines, written and adapted primarily by Perens.[6][7][8] Perens did not base his writing on the "four freedoms" from the Free Software Foundation (FSF), which were only widely available later.[9]

Under Perens' definition, *open source* is a broad software license that makes source code available to the general public with relaxed or non-existent restrictions on the use and modification of the code. It is an explicit "feature" of open source that it puts very few restrictions

on the use or distribution by any organization or user, in order to enable the rapid evolution of the software.[10]

According to Feller et al. (2005), the terms "free software" and "open-source software" should be applied to any "software products distributed under terms that allow users" to use, modify, and redistribute the software "in any manner they see fit, without requiring that they pay the author(s) of the software a royalty or fee for engaging in the listed activities."[11]

Despite initially accepting it,[12] Richard Stallman of the FSF now flatly opposes the term "Open Source" being applied to what they refer to as "free software". Although he agrees that the two terms describe "almost the same category of software", Stallman considers equating the terms incorrect and misleading.[13] Stallman also opposes the professed pragmatism of the Open Source Initiative, as he fears that the free software ideals of freedom and community are threatened by compromising on the FSF's idealistic standards for software freedom.[14] The FSF considers free software to be a subset of open-source software, and Richard Stallman explained that DRM software, for example, can be developed as open source, despite that it does not give its users freedom (it restricts them), and thus does not qualify as free software.[13]

Open-source software development

Development model

In his 1997 essay *The Cathedral and the Bazaar*, open-source influential contributor Eric S. Raymond suggests a model for developing OSS known as the *bazaar* model.[15] Raymond likens the development of software by traditional methodologies to building a cathedral, with careful isolated work by individuals or small groups.[15] He suggests that all software should be developed using the bazaar style, with differing agendas and approaches.[15]

In the traditional model of development, which he called the *cathedral* model, development takes place in a centralized way.[15] Roles are clearly defined.[15] Roles include people dedicated to designing (the architects), people responsible for managing the project, and people responsible for implementation.[15] Traditional software engineering follows the cathedral model.[15]

The bazaar model, however, is different.[15] In this model, roles are not clearly defined.[15] Some proposed characteristics of software developed using the bazaar model should exhibit the following patterns:[16]

Users should be treated as co-developers: The users are treated like co-developers and so they should have access to the source code of the software.[16] Furthermore, users are encouraged to submit additions to the software, code fixes for the software, bug reports, documentation, etc. Having more co-developers increases the rate at which the software evolves.[16] Linus's law states that given enough eyeballs all bugs are shallow.[16] This means that if many users view the source code, they will eventually find all bugs and suggest how to fix them.[16] Some users have advanced programming skills, and furthermore, each user's machine provides an additional testing environment.[16] This new testing environment offers the ability to find and fix a new bug.[16]

Early releases: The first version of the software should be released as early as possible so as to increase one's chances of finding co-developers early.[16]

Frequent integration: Code changes should be integrated (merged into a shared code base) as often as possible so as to avoid the overhead of fixing a large number of bugs at the end of the project life cycle.[16][17] Some open-source projects have nightly builds where integration is done automatically.[16]

Several versions: There should be at least two versions of the software.[16] There should be a buggier version with more features and a more stable version with fewer features.[16] The buggy version (also called the development version) is for users who want the immediate use of the latest features and are willing to accept the risk of using code that is not yet thoroughly tested.[16] The users can then act as co-developers, reporting bugs and providing bug fixes.[16][18]

High modularization: The general structure of the software should be modular allowing for parallel development on independent components.[16]

Dynamic decision-making structure: There is a need for a decision-making structure, whether formal or informal, that makes strategic decisions depending on changing user requirements and other factors.[16] Compare with extreme programming.[16]

The process of Open source development begins with a requirements elicitation where developers consider if they should add new features or if a bug needs to be fixed in their project.[18] This is established by communicating with the OSS community through avenues such as bug reporting and tracking or mailing lists and project pages.[18] Next, OSS developers select or are assigned to a task and identify a solution. Because there are often many different possible routes for solutions in OSS, the best solution must be chosen with careful consideration and sometimes even peer feedback.[18] The developer then begins to develop and commit the code.[18] The code is then tested and reviewed by peers.[18] Developers can edit and evolve their code through feedback from continuous integration.[18] Once the leadership and community are satisfied with the whole project, it can be partially released and user instruction can be documented.[18] If the project is ready to be released, it is frozen, with only serious bug fixes or security repairs occurring.[18] Finally, the project is fully released and only changed through minor bug fixes.[18]

Advantages

Open source implementation of a standard can increase adoption of that standard.[19] This creates developer loyalty as developers feel empowered and have a sense of ownership of the end product.[20]

Moreover, lower costs of marketing and logistical services are needed for OSS.[21] OSS can be a tool to promote a company's image, including its commercial products.[22] The OSS development approach has helped produce reliable, high quality software quickly and inexpensively.[21]

Open source development offers the potential to quicken innovation and create of social value.[23] In France for instance, a policy that incentivized government to favor free open-source software increased to nearly 600,000 OSS contributions per year, generating social value by increasing the quantity and quality of open-source software.[23] This policy also led to an estimated increase of up to 18% of tech startups and a 14% increase in the number of people employed in the IT sector.[23]

OSS can be highly reliable when it has thousands of independent programmers testing and fixing bugs of the software.[16] Open source is not dependent on the company or author that

originally created it.[24] Even if the company fails, the code continues to exist and be developed by its users.[24]

OSS is flexible because modular systems allow programmers to build custom interfaces, or add new abilities to it and it is innovative since open-source programs are the product of collaboration among a large number of different programmers.[16] The mix of divergent perspectives, corporate objectives, and personal goals speeds up innovation.[25]

Moreover, free software can be developed in accordance with purely technical requirements.[26] It does not require thinking about commercial pressure that often degrades the quality of the software.[26] Commercial pressures make traditional software developers pay more attention to customers' requirements than to security requirements, since such features are somewhat invisible to the customer.[26]

Development tools

In open-source software development, tools are used to support the development of the product and the development process itself.[18]

Version control systems such as Centralized Version control system (CVCS) and the distributed version control system (DVCS) are examples of tools, often open source, that help manage the source code files and the changes to those files for a software project in order to foster collaboration.[27] CVCS are centralized with a central repository while DVCS are decentralized and have a local repository for every user.[27] concurrent versions system (CVS) and later Subversion (SVN) and Git are examples of CVCS.[27] The repositories are hosted and published on source-code-hosting facilities such as GitHub.[27]

Open-source projects use utilities such as issue trackers to organize open-source software development. Commonly used bugtrackers include Bugzilla and Redmine.[18]

Tools such as mailing lists and IRC provide means of coordination and discussion of bugs among developers.[18] Project web pages, wiki pages, roadmap lists and newsgroups allow for the distribution of project information that focuses on end users.[18]

Opportunities for participation

[

Contributing

The basic roles OSS participants can fall into multiple categories, beginning with leadership at the center of the project who have control over its execution.[28] Next are the core contributors with a great deal of experience and authority in the project who may guide the other contributors.[28] Non-core contributors have less experience and authority, but regularly contribute and are vital to the project's development.[28] New contributors are the least experienced but with mentorship and guidance can become regular contributors.[28]

Some possible ways of contributing to open-source software include such roles as programming, user interface design and testing, web design, bug triage, accessibility design and testing, UX design, code testing, and security review and testing.[28] However, there are several ways of contributing to OSS projects even without coding skills.[28] For example, some less technical ways of participating are documentation writing and editing, translation, project

management, event organization and coordination, marketing, release management, community management, and public relations and outreach.[28]

Funding is absolutely another terrific way that individuals and organizations choose to contribute to open source projects. Groups like Open Collective provide a means for individuals to contribute monthly to supporting their favorite projects.[29] Organizations like the Sovereign Tech Fund is able to contribute to millions to supporting the tools the German Government uses.[30] The National Science Foundation established a Pathways to Enable Open-Source Ecosystems (POSE) program to support open source innovation.[31]

Industry participation

The adoption of open-source software by industry is increasing over time.[32] OSS is popular in several industries such as telecommunications, aerospace, healthcare, and media & entertainment due to the benefits it provides.[33] Adoption of OSS is more likely in larger organizations and is dependent on the company's IT usage, operating efficiencies, and the productivity of employees.[32]

Industries are likely to use OSS due to back-office functionality, sales support, research and development, software features, quick deployment, portability across platforms and avoidance of commercial license management.[32] Additionally, lower cost for hardware and ownership are also important benefits.[32]

Prominent organizations

Organizations that contribute to the development and expansions of free and open-source software movements exist all over the world.[28] These organizations are dedicated to goals such as teaching and spreading technology.[28] As listed by a former vice president of the Open Source Initiative, some American organizations include the Free Software Foundation, Software Freedom Conservancy, the Open Source Initiative and Software in the Public Interest.[28] Within Europe some notable organizations are Free Software Foundation Europe, open-source projects EU (OSP) and OpenForum Europe (OFE).[28] One Australian organization is Linux Australia while Asia has Open source Asia and FOSSAsia.[28] Free and open source software for Africa (FOSSFA) and OpenAfrica are African organizations and Central and South Asia has such organizations as FLISOL and GRUP de usuarios de software libre Peru.[28] Outside of these, many more organizations dedicated to the advancement of open-source software exist.[28]

Legal and economic issues

Licensing

FOSS products are generally licensed under two types of licenses: permissive licensing and copyleft licensing.[34] Both of these types of licenses are different than proprietary licensing in that they can allow more users access to the software and allow for the creation of derivative works as specified by the terms of the specific license, as each license has its own rules.[34] Permissive licenses allow recipients of the software to implement the author's copyright rights without having to use the same license for distribution.[34] Examples of this type of license include the BSD, MIT, and Apache licenses.[34] Copyleft licenses are different in that they require recipients to use the same license for at least some parts of the distribution of their works.[34] Strong copyleft licenses require all derivative works to use the same license while weak copyleft licenses require the use of the same license only under certain conditions.[34] Examples of this type of license include the GNU family of licenses, and

the MPL and EPL licenses.[34] The similarities between these two categories of licensing include that they provide a broad grant of copyright rights, require that recipients preserve copyright notices, and that a copy of the license is provided to recipients with the code.[34]

One important legal precedent for open-source software was created in 2008, when the Jacobson v Katzer case enforced terms of the Artistic license, including attribution and identification of modifications.[34] The ruling of this case cemented enforcement under copyright law when the conditions of the license were not followed.[34] Because of the similarity of the Artistic license to other open-source software licenses, the ruling created a precedent that applied widely.[34]

Examples of free-software license / open-source licenses include Apache licenses, BSD licenses, GNU General Public Licenses, GNU Lesser General Public License, MIT License, Eclipse Public License and Mozilla Public License.[34]

Legal issues

Several gray areas exist within software regulation that have great impact on open-source software, such as if software is a good or service, what can be considered a modification, governance through contract vs license, ownership and right of use.[34] While there have been developments on these issues, they often lead to even more questions.[34] The existence of these uncertainties in regulation has a negative impact on industries involved in technologies as a whole.[34]

Within the legal history of software as a whole, there was much debate on whether to protect it as intellectual property under patent law, copyright law or establishing a unique regulation.[34] Ultimately, copyright law became the standard with computer programs being considered a form of literary work, with some tweaks of unique regulation.[34]

Software is generally considered source code and object code, with both being protectable, though there is legal variety in this definition.[34] Some jurisdictions attempt to expand or reduce this conceptualization for their own purposes.[34] For example, The European Court of Justice defines a computer program as not including the functionality of a program, the programing language, or the format of data files.[34] By limiting protections of the different aspects of software, the law favors an open-source approach to software use.[34] The US especially has an open approach to software, with most open-source licenses originating there.[34] However, this has increased the focus on patent rights within these licenses, which has seen backlash from the OSS community, who prefer other forms of IP protection.[34]

Another issue includes technological protection measures (TPM) and digital rights management (DRM) techniques which were internationally legally recognized and protected in the 1996 World Intellectual Property Organization (WIPO) Treaty.[34] Open source software proponents disliked these technologies as they constrained end-users potentially beyond copyright law.[34] Europe responded to such complaints by putting TPM under legal controls, representing a victory for OSS supporters.[34]

Economic/business implications

In open-source communities, instead of owning the software produced, the producer owns the development of the evolving software.[35] In this way, the future of the software is open, making ownership or intellectual property difficult within OSS.[35] Licensing and branding can prevent others from stealing it, preserving its status as a public good.[35] Open source software can be

considered a public good as it is available to everyone and does not decrease in value for others when downloaded by one person.[35] Open source software is unique in that it becomes more valuable as it is used and contributed to, instead of diminishing the resource. This is explained by concepts such as investment in reputation and network effects.[35]

The economic model of open-source software can be explained as developers contribute work to projects, creating public benefits.[35] Developers choose projects based on the perceived benefits or costs, such as improved reputation or value of the project.[35] The motivations of developers can come from many different places and reasons, but the important takeaway is that money is not the only or even most important incentivization.[35]

Because economic theory mainly focuses on the consumption of scarce resources, the OSS dynamic can be hard to understand. In OSS, producers become consumers by reaping the rewards of contributing to a project.[35] For example, a developer becomes well regarded by their peers for a successful contribution to an OSS project.[35] The social benefits and interactions of OSS are difficult to account for in economic models as well.[35] Furthermore, the innovation of technology creates constantly changing value discussions and outlooks, making economic model unable to predict social behavior.[35]

Although OSS is theoretically challenging in economic models, it is explainable as a sustainable social activity that requires resources.[35] These resources include time, money, technology and contributions.[35] Many developers have used technology funded by organizations such as universities and governments, though these same organizations benefit from the work done by OSS.[35] As OSS grows, hybrid systems containing OSS and proprietary systems are becoming more common.[35]

Throughout the mid 2000s, more and more tech companies have begun to use OSS.[24] For example, Dell's move of selling computers with GNU/Linux already installed.[24] Microsoft itself has launched a Linux-based operating system despite previous animosity with the OSS movement.[24] Despite these developments, these companies tend to only use OSS for certain purposes, leading to worries that OSS is being taken advantage of by corporations and not given anything in return.[24]

Government uses

While many governments are interested in implementing and promoting open-source software due to the many benefits provided, a huge issue to be considered is cybersecurity.[36] While accidental vulnerabilities are possible, so are attacks by outside agents.[36] Because of these fears, governmental interest in contributing to the governance of software has become more prominent.[36] However, these are the broad strokes of the issue, with each country having their own specific politicized interactions with open-source software and their goals for its implementation.[36] For example, the United States has focused on national security in regard to open-source software implementation due to the perceived threat of the increase of open-source software activity in countries like China and Russia, with the Department of Defense considering multiple criteria for using OSS.[36] These criteria include: if it comes from and is maintained by trusted sources, whether it will continue to be maintained, if there are dependencies on sub-components in the software, component security and integrity, and foreign governmental influence.[36]

Another issue for governments in regard to open source is their investments in technologies such as operating systems, semiconductors, cloud, and artificial intelligence.[36] These

technologies all have implications for global cooperation, again opening up security issues and political consequences.[36] Many countries have to balance technological innovation with technological dependence in these partnerships.[36] For example, after China's open-source dependent company Huawei was prevented from using Google's Android system in 2019, they began to create their own alternative operating system: Harmony OS.[36]

Germany recently established a Sovereign Tech Fund, to help support the governance and maintenance of the software that they use.

Open software movement

History

In the early days of computing, such as the 1950s and into the 1960s, programmers and developers shared software to learn from each other and evolve the field of computing.[37] For example, Unix included the operating system source code for users.[37] Eventually, the commercialization of software in the years 1970–1980 began to prevent this practice.[37] However, academics still often developed software collaboratively.[37]

In response, the open-source movement was born out of the work of skilled programmer enthusiasts, widely referred to as hackers or hacker culture.[38] One of these enthusiasts, Richard Stallman, was a driving force behind the free software movement, which would later allow for the open-source movement.[17] In 1984, he resigned from MIT to create a free operating system, GNU, after the programmer culture in his lab was stifled by proprietary software preventing source code from being shared and improved upon.[17] GNU was UNIX compatible, meaning that the programmer enthusiasts would still be familiar with how it worked.[17] However, it quickly became apparent that there was some confusion with the label Stallman had chosen of free software, which he described as free as in free speech, not free beer, referring to the meaning of free as freedom rather than price.[17] He later expanded this concept of freedom to the four essential freedoms.[17] Through GNU, open-source norms of incorporating others' source code, community bug fixes and suggestions of code for new features appeared.[17] In 1985, Stallman founded the Free Software Foundation (FSF) to promote changes in software and to help write GNU.[17] In order to prevent his work from being used in proprietary software, Stallman created the concept of copyleft, which allowed the use of his work by anyone, but under specific terms.[17] To do this, he created the GNU General Public License (GNU GPL) in 1989, which was updated in 1991.[17] In 1991, GNU was combined with the Linux kernel written by Linus Torvalds, as a kernel was missing in GNU.[39] The operating system is now usually referred to as Linux.[17] Throughout this whole period, there were many other free software projects and licenses around at the time, all with different ideas of what the concept of free software was and should be, as well as the morality of proprietary software, such as Berkeley Software Distribution, TeX, and the X Window System.[40]

As free software developed, the Free Software Foundation began to look how to bring free software ideas and perceived benefits to the commercial software industry.[40] It was concluded that FSF's social activism was not appealing to companies and they needed a way to rebrand the free software movement to emphasize the business potential of sharing and collaborating on software source code.[40] The term open source was suggested by Christine Peterson in 1998 at a meeting of supporters of free software.[17] Many in the group felt the name free software was confusing to newcomers and holding back industry interest and they readily accepted the new designation of open source, creating the Open Source Initiative (OSI) and the OSI definition of what open source software is.[17] The Open Source Initiative's (OSI) definition is now recognized

by several governments internationally as the standard or *de facto* definition.[39] The definition was based on the Debian Free Software Guidelines, written and adapted primarily by Bruce Perens.[41] The OSI definition differed from the free software definition in that it allows the inclusion of proprietary software and allows more liberties in its licensing.[17] Some, such as Stallman, agree more with the original concept of free software as a result because it takes a strong moral stance against proprietary software, through there is much overlap between the two movements in terms of the operation of the software.[17]

While the Open Source Initiative sought to encourage the use of the new term and evangelize the principles it adhered to, commercial software vendors found themselves increasingly threatened by the concept of freely distributed software and universal access to an application's source code, with an executive of Microsoft calling open source an intellectual property destroyer in 2001.[42] However, while free and open-source software (FOSS) has historically played a role outside of mainstream private software development, companies as large as Microsoft have begun to develop official open source presences on the Internet.[42] IBM, Oracle, and State Farm are just a few of the companies with a serious public stake in today's competitive open source market, marking a significant shift in the corporate philosophy concerning the development of FOSS.[42]

Future

The future of the open source software community, and the free software community by extension, has become successful if not confused about what it stands for.[24] For example, Android and Ubuntu are examples milestones of success in the open source software rise to prominence from the sidelines of technological innovation as it existed in the early 2000s.[24] However, some in the community consider them failures in their representation of OSS due to issues such as the downplaying of the OSS center of Android by Google and its partners, the use of an Apache license that allowed forking and resulted in a loss of opportunities for collaboration within Android, the prioritization of convenience over freedom in Ubuntu, and features within Ubuntu that track users for marketing purposes.[24]

The use of OSS has become more common in business with 78% of companies reporting that they run all or part of their operations on FOSS.[24] The popularity of OSS has risen to the point that Microsoft, a once detractor of OSS, has included its use in their systems.[24] However, this success has raised concerns that will determine the future of OSS as the community must answer questions such as what OSS is, what should it be, and what should be done to protect it, if it even needs protecting.[24] All in all, while the free and open source revolution has slowed to a perceived equilibrium in the market place, that does not mean it is over as many theoretical discussions must take place to determine its future.[24]

Comparisons with other software licensing/development models

Closed source / proprietary software

Open source software differs from proprietary software in that it is publicly available, the license requires no fees, modifications and distributions are allowed under license specifications.[43] All of this works to prevent a monopoly on any OSS product, which is a goal of proprietary software.[43] Proprietary software limits their customers' choices to either committing to using that software, upgrading it or switching to other software, forcing customers to have their software preferences impacted by their monetary cost.[43] The ideal case scenario for the proprietary

software vendor would be a lock-in, where the customer does not or cannot switch software due to these costs and continues to buy products from that vendor.[43]

Within proprietary software, bug fixes can only be provided by the vendor, moving platforms requires another purchase and the existence of the product relies on the vendor, who can discontinue it at any point.[38] Additionally, proprietary software does not provide its source code and cannot be altered by users.[17] For businesses, this can pose a security risk and source of frustration, as they cannot specialize the product to their needs, and there may be hidden threats or information leaks within the software that they cannot access or change.[17]

Free software

Under OSI's definition, open source is a broad software license that makes source code available to the general public with relaxed or non-existent restrictions on the use and modification of the code.[44] It is an explicit feature of open source that it puts very few restrictions on the use or distribution by any organization or user, in order to enable the rapid evolution of the software.[44]

Richard Stallman, leader of the Free software movement and member of the free software foundation opposes the term open source being applied to what they refer to as free software.[13] Although he agrees that the two terms describe almost the same category of software, Stallman considers equating the terms incorrect and misleading.[13] He believes that the main difference is that by choosing one term over the other lets others know about what one's goals are: development (open source) or a social stance (free software).[45] Nevertheless, there is significant overlap between open source software and free software.[13] Stallman also opposes the professed pragmatism of the Open Source Initiative, as he fears that the free software ideals of freedom and community are threatened by compromising on the FSF's idealistic standards for software freedom.[45] The FSF considers free software to be a subset of open-source software, and Richard Stallman explained that DRM software, for example, can be developed as open source, despite how it restricts its users, and thus does not qualify as free software.[13]

The FSF said that the term open source fosters an ambiguity of a different kind such that it confuses the mere availability of the source with the freedom to use, modify, and redistribute it.[13] On the other hand, the term free software was criticized for the ambiguity of the word free, which was seen as discouraging for business adoption, and for the historical ambiguous usage of the term.[45]

Developers have used the alternative terms *Free and Open Source Software* (FOSS), or *Free/Libre and Open Source Software* (FLOSS), consequently, to describe open-source software that is also free software.[28]

Source-available software

Software can be distributed with source code, which is a code that is readable.[46] Software is source available when this source code is available to be seen.[46] However to be source available or FOSS, the source code does not need to be accessible to all, just the users of that software.[46] While all FOSS software is source available because this is a requirement made by the Open Source Definition, not all source available software is FOSS.[46] For example, if the software doesn't meet other aspects of the Open Source Definition such as permitted modification or redistribution, even if the source code is available, the software is not FOSS.[46]

Open-sourcing

A recent trend within software companies is open sourcing, or transitioning their previous proprietary software into open source software through releasing it under a open-source license.[47][48] Examples of companies who have done this are Google, Microsoft and Apple.[47] Additionally, open sourcing can refer to programming open source software or installing open source software.[48] Open sourcing can be beneficial in multiple ways, such as attracting more external contributors who bring new perspectives and problem solving capabilities.[47] The downsides of open sourcing include the work that has to be done to maintaining the new community, such as making the base code easily understandable, setting up communication channels for new developers and creating documentation to allow new developers to easily join.[47] However, a review of several open sourced projects found that although a newly open sourced project attracts many newcomers, a great amount are likely to soon leave the project and their forks are also likely to not be impactful.[47]

Other

Other concepts that may share some similarities to open source are shareware, public domain software, freeware, and software viewers/readers that are freely available but do not provide source code.[17] However, these differ from open source software in access to source code, licensing, copyright and fees.[17]

Society and culture

Demographics

Despite being able to collaborate internationally, open source software contributors were found to mostly be located in large clusters such as Silicon Valley that largely collaborate within themselves.[49] Possible reasons for this phenomenon may be that the OSS contributor demographic largely works in software, meaning that the OSS geographic location is closely related to that dispersion and collaborations could be encouraged through work and social networks.[49] Code acceptance can be impacted by status within these social network clusters, creating unfair predispositions in code acceptance based on location.[50] Barriers to international collaboration also include linguistic or cultural differences.[51] Furthermore, each country has been shown to have a higher acceptance rate for code from contributors within their country except India, indicating a bias for culturally similar collaborators.[51]

In 2021, the countries with the highest open source software contributions included the United States, China, Germany, India, and the UK, in that order.[49] The counties with the highest OSS developers per capita from a study in 2021 include, in order, Iceland, Switzerland, Norway, Sweden, and Finland, while in 2008 the countries with top amount of estimated contributors in SourceForge were the United States, Germany, United Kingdom, Canada and France.[49][51] Though there have been several studies done on the distribution and contributions of OSS developers, this is still an open field that can be measured in several different ways.[51] For instance, Information and communication technology participation, population, wealth and proportion of access to the internet have been shown to be correlated with OSS contributions.[51]

Although gender diversity has been found to enhance team productivity, women still face biases while contributing to open source software projects when their gender is identifiable.[52] In 2002, only 1.5% of international open-source software developers were women, while women made up 28% of tech industry roles, demonstrating their low representation in the software

field.[53] Despite OSS contributions having no prerequisites, this gender bias may continue to exist due to the common belief of contributors that gender shouldn't matter, and the quality of code should be the only consideration for code acceptance, preventing the community from addressing the systemic disparities in female representation.[38] However, a more recent figure of female OSS participation internationally calculated across 2005 to 2021 is 9.8%, with most being recent contributors, indicating that female participation may be growing.[54]

Motivations

[

There are many motivations for contributing to the OSS community.[28] For one, it is an opportunity to learn and practice multiple skills such as coding and other technology related abilities, but also fundamental skills such as communication and collaboration and practical skills needed to excel in technology related fields such as issue tracking or version control.[28] Instead of learning through a classroom or a job, learning through contributing to OSS allows participants to learn at their own pace and follow what interests them.[28] When contributing to OSS, you can learn the current industry best practices, technology and trends and even have the opportunity to contribute to the next big innovation as OSS grows increasingly popular within the tech field.[28] Contributing to OSS without payment means there is no threat of being fired, though reputations can take a hit.[28] On the other hand, a huge motivation to contribute to OSS is the reputation gained as you grow your public portfolio.[28]

Disparities

Even though programming was originally seen as a female profession, there remains a large gap in computing.[55] Social identity tends to be a large concern as women in the tech industry face insecurity about attracting unwanted male attention and harassment or being unfeminine in their technology knowledge, having a large impact on confidence.[38] Some male tech participants make clear that they believe women fitting in within the culture is impossible, furthering the insecurity for women and their place in the tech industry.[52] Additionally, even in a voluntary contribution environment like open source software, women tend to end up doing the less technical aspects of projects, such as manual testing or documentation despite women and men showing the same productivity in OSS contributions.[52] Explicit biases include longer feedback time, more scrutinization of code and lower acceptance rate of code.[52] Specifically in the open-source software community, women report that sexually offensive language is common and the women's identity as female is given more attention that as an OSS contributor[38] Bias is hard to address due to the belief that gender shouldn't matter, with most contributors feeling that women getting special treatment is unfair and success should be dependent on skill, preventing any changes to be more inclusive.[38]

Adoption and application

Key projects

Open source software projects are built and maintained by a network of programmers, who may often be volunteers, and are widely used in free as well as commercial products.[56]

Unix: Unix is an operating system created by AT&T that began as a precursor to open source software in that the free and open-source software revolution began when developers began trying to create operating systems without Unix code.[24] Unix was created in the 1960s, before the commercialization of software and before the concept of open source software was necessary, therefore it was not considered a true open source software project.[24] It started as a

research project before being commercialized in the mid 1980s.[24] Before its commercialization, it represented many of the ideals held by the Free and Open source software revolution, including the decentralized collaboration of global users, rolling releases and a community culture of distaste towards proprietary software.[24]

BSD: Berkely Software Distribution (BSD) is an operating system that began as a variant of Unix in 1978 that mixed Unix code with code from Berkely labs to increase functionality.[24] As BSD was focused on increasing functionality, it would publicly share its greatest innovations with the main Unix operating system.[24] This is an example of the free public code sharing that is a central characteristic of FOSS today.[24] As Unix became commercialized in the 1980s, developers or members of the community who did not support proprietary software began to focus on BSD and turning it into an operating system that did not include any of Unix's code.[24]The final version of BSD was released in 1995.[24]

GNU: GNU is a free operating system created by Richard Stallman in 1984 with its name meaning Gnu's Not Unix.[24] The idea was to create a Unix alternative operating system that would be available for anyone to use and allow programmers to share code freely between them.[24] However, the goal of GNU was not to only replace Unix, but to make a superior version that had more technological capabilities.[24] It was released before the philosophical beliefs of the Free and Open source software revolution were truly defined.[24] Because of its creation by prominent FOSS programmer Richard Stallman, GNU was heavily involved in FOSS activism, with one of the greatest achievements of GNU being the creation of the GNU General Public License or GPL, which allowed developers to release software that could be legally shared and modified.[24]

Linux: Linux is an operating system kernel that was introduced in 1991 by Linus Torvalds.[24] Linux was inspired by making a better version of the for profit operating service Minux.[24] It was radically different than what other hackers were producing at the time due to it being totally free of cost and being decentralized.[24] Later, Linux was put under the GPL license, allowing people to make money with Linux and bringing Linux into the FOSS community.[24]

Apache: Apache began in 1995 as a collaboration between a group of developers releasing their own web server due to their frustration with NCSA HTTPd code base.[24] The name Apache was used because of the several patches they applied to this code base.[24] Within a year of its release, it became the worldwide leading web server.[24] Soon, Apache came out with its own license, creating discord in the greater FOSS community, though ultimately proving successful.[24] The Apache license allowed permitted members to directly access source code, a marked difference from GNU and Linux's approaches.[24]

Extensions for non-software use

While the term open source applied originally only to the source code of software, it is now being applied to many other areas such as open-source ecology, a movement to decentralize technologies so that any human can use them.[13][57] However, it is often misapplied to other areas that have different and competing principles, which overlap only partially.[38]

The same principles that underlie open-source software can be found in many other ventures, such as open source, open content, and open collaboration.[58][3]

This "culture" or ideology takes the view that the principles apply more generally to facilitate concurrent input of different agendas, approaches, and priorities, in contrast with more centralized models of development such as those typically used in commercial companies.[15]

Value

More than 90 percent of companies use open-source software as a component of their proprietary software.[59] The decision to use open-source software, or even engage with open-source projects to improve existing open-source software, is typically a pragmatic business decision.[60][61] When proprietary software is in direct competition with an open-source alternative, research has found conflicting results on the effect of the competition on the proprietary product's price and quality.[62]

For decades, some companies have made servicing of an open-source software product for enterprise users their business model. These companies control an open-source software product, and instead of charging for licensing or use, charge for improvements, integration, and other servicing.[63] Software as a service (SaaS) products based on open-source components are increasingly common.[64]

Open-source software is preferred for scientific applications, because it increases transparency and aids in the validation and acceptance of scientific results.[65]