Open Source Software

Motivations and Challenges

Edwin Chan
Ryan Damm
Sean Mikalson
Ho Wai Yung

I. INTRODUCTION

A. *Brief History*

While the tradition of sharing technological knowledge has existed in human history ever since humans started developing the most basic tools and technologies, the free sharing of software technologies seen today, referred to as open source, began with the birth of the internet. The internet provided a mechanism where source code could be easily transferred and accessed by a large number of people in various locations around the world, rather than source code sitting on only one person's local machine. This allowed large numbers of people to contribute and collaborate on a single software project.

In addition to the emergence of the internet, two projects significantly helped the open source movement gain momentum. First, was the GNU project and the Free Software Foundation founded by Richard Stallman, which aimed at creating a free operating system as well as outlining that software released under the General Public License prohibits proprietization of the software and preserves the freedoms of users to freely distribute and modify the software as they desire. GNU and FSF helped establish the fundamental freedoms that open source, in general, tries to promote and preserve which are the freedom to use, modify, redistribute, and distribute your modified copy of the software. The second project establishing the open source movement was the Computer Science Research Group which aimed at freeing the Unix operating system from AT&T copyrighted code as well as adding improvements and applications to the Unix operating system. The group utilized the internet to engage Unix hackers around the world to maintain and improve the system. The CSRG pioneered the distributed development nature that is seen in every open source project today.

In the early 1990's the two groups laying the foundation for open source development released their operating systems, with Linus Torvalds teaming up with the GNU project to release the GNU/Linux operating system and the CSRG releasing the improved Unix operating system as 386BSD operating system. These two projects represent the first two major open source projects and encouraged free sharing and collaboration of software projects. The event that completes the evolution of the open source movement from its beginnings to how we see it today was the naming of the movement as Open Source coinciding with the release of Netscape's source code for its Navigator internet browser, coincidentally paying homage to the very thing that gave open source its beginnings, the internet. Around the release of Navigator's source code, a strategy session was held by a group of people involved in free software projects; they created and adopted the term open source, and in effect, legitimized the process of the open source development model in the world of software technologies. The term open source was advantageous as the business world was hesitant in installing 'free' software technologies, a hesitation that was avoided by adopting the term Open Source [0].

B. *The Nature of Open Source Development*

Open source development is the development of a project in an open and transparent manner where a project's source code is freely distributed so that the general public may participate in that project's development. As a result, software developed in a public setting is subject to the implementation and evaluation of many people with diverse perspectives and skills. This kind of crowdsourcing and collaboration is an essential feature of open source projects and in some cases, it may be argued that this results in higher quality projects. Conversely, there are many challenges that open source projects face and even the same aspect (e.g. the diverse nature of the development team) can prove to be an advantage for one project while being a challenge for another. This paper discusses the many different aspects involved in software development that are affected by the open source development philosophy, specifically how open source development can benefit or hinder these aspects during development process.

The first category of aspects considered in this paper are the various technical aspects within software projects including code quality, availability, security, standards development, flexibility, innovative solutions and licensing issues. These more technical aspects of software development can be greatly enhanced in an open source community. Code quality, for example, can be better in open source projects due to the necessity to make the software more modular so that new contributions to the project can be easily included. However, code quality can be decreased in open source as well, if the project lacks a clear direction and goal resulting in code that does not work together to achieve a common goal; both how open source benefits and harms aspects like code

quality will be discussed. Another technical challenge that will be discussed is the technical legal issues that open source projects face; choosing the correct license to release the software under can affect how well the software is supported in the future and how well the project attracts more and better quality developers affecting the technical quality of the software

The second category of aspects affected by open source development are human aspects. There are many human motivations to contribute to open source projects, such as having fun, learning new things as well as increasing your reputation and thus your worth. These motivations can also apply to companies as they can benefit in much in the same way as individuals; they can increase the skill in their workforce as well as increase their reputation. There are, however, human challenges when contributing to open source software. These can include things like social hierarchies, policies, funding, and differences in opinion which affect the motivations and teamwork of developers in a negative way, as well as the creation and uniformity of a design plan.

As we will demonstrate in this paper, the technical and human aspects that come with open source software development affect its production. Depending on the benefits and drawbacks, open source development may be beneficial to some projects, while harmful to others. These attributes must be considered when determining what design philosophy to apply to a project for an optimal scenario.

II. Technical Aspects

A. Motivations

1) Code Quality

Open Source software development consists of code contributed by countless developers, each with varying degrees of involvement in a project. The nature of Open Source development allows for individuals to work independently, at their own pace, and in their own way. To facilitate the sharing and implementation of new code, each person's code must be compatible with existing code. To distribute tasks to the community, projects must be designed such that parts can be worked on individually. To continuously improve the source code with such a great number of simultaneous contributors, it must be possible to swap code in and out of a project without affecting the rest of the project [3]. All these factors lead to one result: highly modular and portable code.

Modular code is a significant benefit when it comes to code quality [2]. With authors working in parallel, parts of the source code can be swapped out for improved solutions. At the same time features can be included or excluded from releases or distributions with relative ease, allowing for more stable releases as well as quicker releases for quicker feedback. Modular code is also fundamental to peer review, because review and testing can be done on each piece of code without worrying about highly coupled code. It becomes a much simpler task to organize and identify code contributed by different developers if the code remains modular.

In addition to the modular nature of Open Source development, transparency and accessibility of the source code further enhances the quality of any given code [1]. Code can be reviewed or tested by any individual immediately after being published, allowing bugs to be found by the public before it is even integrated or deployed. Testers that find bugs are often users of the software themselves, being more motivated to report or even fix the bugs themselves. While quality of code submissions may vary depending on the individual, only thoroughly scrutinized and tested code makes it to the final releases.

2) Availability

Returning to the accessibility of code briefly mentioned previously, Open Source describes itself in that the source code is available to all to view, and in some cases modify or extend. Source code is often uploaded on public repositories online, providing access to any interested individual. The openness of the project invites contributors as well as keen users facilitating as testers. With write-access to the source code, bug fixes can be facilitated as direct changes to the code.

3) Security

Security is a hot debate for Open Source projects when being compared to commercial projects. While proponents of commercial projects argue that security is improved through restricted access to source code, large commercial projects are often just as susceptible to exploits and hacks. Open Source development has the benefit of what's often described as Linus's Law, which states "given enough eyeballs, all bugs are shallow." As opposed to the limited testers in a commercial environment, Open Source code is available to anyone and everyone to freely access. Given this openness, it is logical to believe that bugs are discovered quicker. There is some research entailing the irrelevance of commercial or Open Source development to the release time of patches or fixes to bugs [4]. Schryen concludes that even amongst the same commercial vendor there are drastic differences in patch releases for different software offerings, and that patch release times are much more correlated to the vendor's policy regarding such patches. Given that Open Source code is reviewed and tested by a much larger community than commercial vendors can procure, it is not shocking at all to find that security in Open Source projects are comparable, if not arguably better, than their commercial counterparts.

4) Standard Development

Open Source projects often prefer open standards, with added support for proprietary standards. Choosing open standards benefits individual users as well as companies deploying the software. The main benefits are compatibility and cost. Open standards, as the name implies, has a degree of

openness which proprietary formats cannot provide. Proprietary formats are designed to work only with the software using them, and as such are not as compatible with software provided by third-party developers. The closed nature of proprietary standards also complicates the development process of any software needing to support the standards. From a cost perspective, it is very simple to see how closed standards can restrict choice of the user or company using them. For example, a document created with a proprietary format can lose formatting or fail to open when using third-party software. Anyone sharing documents with the user or within the company would need to use the same commercial software to guarantee full compatibility and support.

5) Flexibility

Open Source projects offer flexibility which can seldom be matched by commercial developments. While commercial products must adhere strictly to a profitable design and release plan, Open Source projects can be modified or even split into separate projects to reflect the community's will. The modular nature previously mentioned gives Open Source code greater flexibility during development as well as releases. From a broader perspective, Open Source projects are not as subject to resource limitations, implying increased flexibility [3].

6) Innovation

Innovation is a product of modular code, availability, and flexibility. Modular code simplifies the process of creating and integrating new code. Availability of source code invites anyone to join the project, bringing their own ideas along. Flexibility of the project allows any contributor an opinion towards code improvement, project direction, features and scope, etc. Developers are left to innovate freely without restrictions inherent in commercial development environments. Ideas receive feedback from the community, and good ideas receive support. A well-supported idea may make it to release, allowing the innovator to utilize a desired feature.

7) Little to no costs for development

While cost is perhaps the most influential factor in commercial development, it is a much less apparent restriction in Open Source development. Contributors freely volunteer their time and expertise, for non-monetary rewards such as recognition by peers in the community or improvements to the software they themselves use [3]. Open Source projects are often started because of the desires and needs of users, which cannot be satiated through commercial means. As such, developers are much more willing to devote time into a cause of interest. The sheer amount of developers able to contribute also reduces cost of a project, without dramatically affecting quality. The resulting software is then made available freely to the community, as a concerted effort by the community itself.

B. Challenges

In many cases, not all technical aspects of open source development are improved relative to proprietary development. First, quality is often reduced due to the inability to control large development groups. Second, bugs may still be developed due to a lack of useful official documentation. Finally, licensing decisions often require the software to be more proprietary simply to motivate developers and compete in a commercial market. Indeed, there are certainly many developmental pitfalls that open source development often has difficulty addressing due to its reliance on large groups of free developers.

1) Quality

Although, as mentioned above, open source projects can result in high quality code there are circumstances and open source projects where code quality is a major issue. This is due to the loss of a "single-minded focus" [5] due to distributed development. For example, one of the many features of open source development is the idea that more people will review the code. In practice, however, most people will "just look at the readme files" [5] rather than do an in-depth code evaluation. The other major issue when relying on this method of confirmation is the lack of familiarity with the software. The reviewers usually do not fully understand all the functionality of the software, and are not able to do an in-depth review. Reviewers will also often not have the training to know specifically what they are looking for.

2) Bugs

In terms of generation of bugs, both open and closed source have similar bug development during the early stages of a project [6]. This would likely be due to a balance between open source projects having a more interdependent development team during the project's infant stages, while at the same time having less documentation than a proprietary development environment would create. Smaller interdependent teams often develop better code thanks to communication and discussion, which reduces bugs. However, they often have less interest in developing thorough documentation due to a more cowboy coded approach, which introduces design and architecture inconsistencies resulting in a larger number of bugs that can quickly become overwhelming for the contributors to the open source project.

3) Licensing

In terms of licenses, they must be chosen carefully for open source project to ensure that the development team continues to be motivated. Open source projects do not have the financial resources that commercial software companies have to keep their developers motivated. Thus, open source projects face the challenge of choosing licences that create incentives and motivations for developers such as licences that appeal to their ideologies and beliefs or licences that do not demand significant efforts by developers to ensure compliance with it. Without proper incentive allowed by the licence, fewer developers will expend as much effort, and the remaining developers will likely be low quality [7]. This can be a major concern when software projects are competing with projects developed with a commercial license as these instances of

commercial developments often use smaller teams with salaries [7] avoiding the difficulties of incentivizing their developers that open source projects face.

When competing across the market, open source projects must often consider the licenses of their competitors. In a commercial market, choosing an open source license requires that license to be more restrictive that help the project maintain control of the development process in order to compete with commercially developed products that already have complete control of their development process [7]. This is often cause for concern because fewer third parties will integrate with open source software using restricted licenses due to less benefits going towards the third party as well as risking scaring away potential contributors due to the restrictive nature of the licence; open source projects must continually balance the degree of openness and restriction in the licences they use when competing in a commercial marketplace.

III. HUMAN ASPECTS

A. Motivations

1) Personal Benefit

Upon discovering who the contributors of open source projects are, mainly Software Engineers and Students (53.4%) with an average age of 27.1 years who contribute on a primarily voluntary basis [8], the question of what motivates these individuals to contribute to projects and what benefits they receive as they are not being compensated for their effort in traditional ways, i.e. getting paid directly for your effort, arises.

A main motivation for developers to start developing open source software is need. For instance, a developer who uses an open source program needs a feature that is currently not implemented or perhaps there is an unfixed bug in the software that the developer needs to be fixed in order to continue using the open source project, the developer might spend time and effort developing the new feature or a patch for the unfixed bug. Immediately, upon completion of the feature or patch, the developer gains some personal benefit from the open source project by solving a problem that the developer had. This example shows the main benefits of open source projects for individual: first, they solve problems people have and distribute these solutions at no or limited cost to the consumer and second, projects allow people to solve their own, unique problems easily and effectively by building on existing code and adding an extra feature. That is, open source projects are aimed at solving problems and provides the foundation and flexibility to motivate developers to change and modify open source code for their own needs and purposes.

However, while allowing developers to obtain and modify open source projects is a major benefit the open source projects provide, these developers have not contributed to open source as of yet. They still need to publish their new feature or patch to the public open source project in order to be considered contributors to the project. Publishing the new feature or patch, thus contributing to open source, comes with non-monetary personal benefits, building on the original benefits of developing software to solve a personal problem. By contributing high quality software to an open source project can be used to signal the developers worth and skill to a large number of software developers also contributing to open source projects. This signaling by contributors builds their reputation as a supplier of high quality software and results in higher skill prices for themselves [8]. Related to building reputations and being able to demand higher skill prices, peer recognition can also act as a personal benefit when a developer contributes to an open source project. Developers may have the desire for others to admire their coding style or admire the elegance of an algorithm/solution to a tricky and difficult problem, and by contributing code to projects the developer has the opportunity to gain this positive feedback from fellow developers [8].

There will be inevitable cases of criticism of code contributed to open source projects, in lieu of the positive feedback and recognition that all developers crave from their respected peers. Rather than this being a disincentive to contribute, this availability of criticism actually provides important and valuable learning opportunities for the community members. Within open source communities there is a vast diversity of people who are contributing, all with various skill sets, experiences, and methodologies. This diversity within the community allows members to learn from experts or learn new solutions. In general, this allows for the free and efficient transmission of knowledge and skills between community members. The benefit of learning is huge in open source projects and is a main reason why many young developers and students get involved in interesting open source projects. The FLOSS-EU survey done in 2002, indicates that 78.9% of the people who joined an open source community did so to learn and develop new skills and 49.8% joined to share knowledge and skills [8]. These statistic indicates nearly everyone in the community benefits from learning new skills, even those that are perceived as experts by their peers, and also that community is committed to sharing knowledge and skills, creating the environment necessary for learning to take place.

2) Human Capital Development for Companies

While the learning environment described above definitely benefits the individuals participating in open source projects, it can also benefit society and companies as well. Companies that leverage the learning environment within open source projects can develop the skills of their employees in a way that is less costly and more effective than traditional, formal training methods. By developing their workforce a company can achieve an adaptive and flexible character, allowing it to respond to an uncertain and dynamic world, via its developed skills and knowledge of their employees, better than its competitors.

As discussed before, developers contributing to open source projects are given the opportunity to have their code reviewed by their peers. This can result in both admiration and criticism and in both cases results in learning opportunities for the community. Companies can leverage open source projects to help develop advanced skills of a company's workers if they allow them to contribute to open source projects. While basic skills may be better learned in a classroom, it has been found that advanced skills are better learned by doing projects [9]. An explanation for this may be that advanced skills require practical experience and knowledge that cannot be acquired through learning theoretical skills from a book. Advanced skills are far more complex and as a result require experience. There is also an indication that advanced skills are better learned in open source projects rather than closed source projects [9]. Participation by company employees in open source projects allows for them to acquire the diverse experiences open source provides in addition to the experience acquired in their jobs. The result is developers that have highly developed advanced skills and makes the company more flexible, adaptive and innovative than their competitors. By allowing their employees to participate in open source projects, companies can develop their human capital and thus provide their customers with superior solutions within proprietary software.

The human capital development has benefits outside of the company as well. When human capital is developed in such a way, say within the boundaries of a particular society, then that society becomes more adaptive, flexible and innovative as well, giving itself a competitive advantage over its competitors. The society may become a leader within software development all through encouraging and have companies leverage the human capital development open source software provides.

3) Organization Structure for Collaboration

In industries that rely on knowledge for innovation and product creation, collaboration and the sharing of knowledge can be used to foster environments that increase creativity. In addition to how collaboration grows and shares the knowledge base thus fostering creativity, it has direct economic benefits as well. Collaboration can reduce risk, move up product delivery and minimize cost of development [10]. The problem for companies is to structure themselves in a way that is conducive to collaboration.

One organizational structure that is geared towards collaboration is Actor-Orientated Organizations where control and coordination are the responsibility of all the organizational members (actors) themselves, as opposed to a hierarchical structure where control and coordination resides with a small group of managers and top executives [10]. When control and coordination are the responsibility of all members of the organization, this results in the ability for members to self-organize, connecting and collaborating with the people they need to very rapidly. When the task is complete and the self-organized group accomplishes their purpose the members have the flexibility to disband and form connections/collaborate elsewhere with other people. The members of an actor-oriented organization do not have the difficulties of climbing the hierarchical ladder only to move two feet to the right and then climb down to reach the person next to them.

The result is a highly flexible, dynamic and innovative organization that, because of the diverse perspectives brought together through collaboration, is able to respond to a constantly changing world and marketplace. Linux, and most likely many other open source projects, demonstrate actor-orientated characteristics and structure [10]. As a community comprised of individuals there is no centralized leadership that determine the direction of Linux as a whole. There may be well-defined leadership within a specific distribution that enforces a specific vision but at no time is the Linux community restrained from self-organizing then forking off current Linux distributions to follow a new vision shared by that new community. This self-organization and connection is established by Linux through the use of discussion boards, blogs and mailing lists where members of the community can engage in discussions with fellow members, learning about others ideas and interests, resulting in the formation of relationships that develop into self-organized groups sharing a common interest and goal [10]. The Linux case demonstrates that the intrinsic structure and nature of open source projects is actor-orientated and fosters collaboration. This structure that the open source movement embodies so successfully shows how organizations can adapt to a world where success is dependent on innovation, speed and flexibility, i.e. where success is dependent on collaboration.

B. Challenges

Open-sourced software projects have roughly the same social problems as the academic based projects since they do not have a desire for profit, but rather the idea of researching a particular problem or idea to satisfy one's curiosity. The social problems that academic projects have are smaller due to the fact the research they do is done in a professional setting while open-source software have these problems magnified due to the open-source orientation of the projects.

1) Politics within open source software

As with any open-source based software, there are many developers that have their own ideas and opinions on how the software should be developed or features they like to see implemented in the project. They are driven by their competitive motives of status and reputation [11]. This in turn leads to disagreements and unnecessary tension in the development of the open-source software.

2) Elitism and accessibility in open-source software

With the development of any particular piece of software, the more lines of code, the harder it is to understand it. Often there is no commenting in the code, leading to accessibility issues in developing the open source project. The contributors

who have the ability to modify and understand the code tend to be people who were at the beginning of the development of the project or people that sacrifice a lot of time in order understand the program [12].

With that in mind, the people who develop the source code and the people who took the time to understand and modify the source code would sometimes consider themselves elites. They would use this power to form social groups that have been known to have shown unprofessional and unruly behaviour in treating other people in their understanding of the software [13]. Luckily, this is has changed drastically over the years and is starting to mature, but it still has room for improvement.

3) Group dynamics in open-source software

Within that group of developers, favouritism is one of the major problems in the development of the software since it can demotivate people if their work is not acknowledged. Any incantation of favouritism from the leader of open-source project would cause internal strife, distrust amongst the group members, destroy initiatives, and undermines the moral of the group. [13]

In an ironic sense, if there is no core group for the open-source software, the development of the software would not progress as quickly. A truly open-source software has been described as a more democracy-based development but is known to stagnant once a project becomes very popular and more developers are eager to provide their own expertise into the project. But with a more authoritative structure, updates would occur more frequently since there is much less bureaucracy in the approval process. [12]

But with any open-source software development, they suffer the same issues as any other big organizations once they become too large or have too much critical mass. They stop being creative and the development of the software becomes stagnant [12]. The project would have a long approval list in order to implement a new feature onto the software.

4) Fragmentation

Open-source software very often suffers from having too many forks or similar versions of the same software. If a group disproves a change in the actual source code, they would often create their own version that implements the features that they want. There is no sense of unity and the original focus is lost with the creation of splinters that occurred from the original open-source program. [12]

5) A New Translation

With open source becoming more mainstream, the development of open-source software becomes more complex. The ability to keep updating requires huge amount of time and money, often times this leads to a business hybrid model of a paid version and a free version of the open source software [14]. Open-source software development has become more of a hybrid business model where the definition of open-source is interpreted in many different ways for businesses. One model is selling the latest version of their software and that version trickles down to the free version [15]. With the ever evolving and rapidly changing landscape of open-source development, open-source development is starting to mature, thus leading to a redefinition of what is called open-source development.

IV. CONCLUSION

Having examined the motivations and challenges of both technical and human aspects in open source development, perhaps the most obvious conclusion is the inconclusiveness of it all. In discussing the technical aspects of open source projects, some motivations included: code quality, modular code, security, and innovation. However, many of these motivations were also identified as challenges themselves, or having a negative impact on development. For example, security is improved because "given enough eyeballs, all bugs are shallow." At the same time, security suffers because of the availability of the code base to hackers. A similar occurrence can be found in the human aspects of open source development, wherein developers are striving for recognition which may lead to elitism.

Despite both motivations and challenges being reasoned and justified, there remains much conflict in the cases presented. A possible and likely explanation is the lack of substantial and in-depth research into a highly varied and complex topic. Much of the existing research attempts to generalize both open source and proprietary development methods, without going further to compare personality traits, work environment, development team size, project goals, etc. The great impact of these factors can easily be seen in another research, where patching behavior deviated immensely despite comparing projects within the same company.

Without the lacking substantial research to support a definitive conclusion relating to the pros and cons of open source development, it would be unwise to blindly choose open source development over proprietary methods, or vice versa. Each motivation or challenge, along with their reasons for being what they are, should be taken into consideration. As presented, there are many reasons why people contribute to open source projects, but there are also enough challenges to sway others away. Neither development style can be touted as the best, but open source development is continuously evolving and continues to interest many developers.

REFERENCES

[0] Wikipedia Article, "Open Source Movement", http://en.wikipedia.org/wiki/Open-source_movement

[1] Peeling and Satchell, "Analysis of the Impact of Open Source Software", QinetiQ, 2001

[2] Gang Peng et al., "Modularity and inequality of code contribution in open source software development", System Science (HICSS), 2012 45th Hawaii International Conference on , pp.4505-4514, Jan 2012

[3] Muegge S. and Milev R., "Measuring modularity in open source code bases", Open Source Business Resource, Apr 2009

[4] Schryen, G., "A comprehensive and comparative analysis of the patching behavior of open source and closed source software vendors", IT Security Incident Management and IT Forensics, 2009. IMF '09. Fifth International Conference on , pp.153-168, Sept 2009

[5] Lawton, G., "Open source security: opportunity or oxymoron?", Computer , vol.35, no.3, pp.18-21, Mar 2002

[6] Zhou, Y. and Davis, J., "Open source software reliability model: an empirical approach", ACM Sigsoft Software Engineering Notes, vol.30, no.4, pp.1-6, 2005

[7] Välimäki, M. and Oksanen, V., "Evaluation of open source licensing models for a company developing mass market software", Law and Technology, 2002

[8] Sauer, R., "Why develop open-source software? The role of non-pecuniary benefits, monetary rewards, and open-source licence type", Oxford Review of Economic Policy, vol.23, no.4, 2007

[9] Mehra, A., Mookerjee, V., "Human capital development for programmers using open source software", MIS Quarterly, vol.36, no.1, 2012

[10] Øystein D. Fjeldstad et al., "The architecture of collaboration", Strategic Management Journal, vol.33, 2012

[11]http://nicomedia.math.upatras.gr/Free-OpenSource/Hertel_etal_MotivationInOSS_Survey.pdf

[12] Bezroukov, N., "Open source software development as a special type of academic research: critique of vulgar raymondism", First Monday [Online], vol.4, no.10, Oct 1999

[13] Bezroukov, N., "A second look at the cathedral and the bazaar", First Monday [Online], vol.4, no.12, Dec 1999

[14] http://www.lambdassociates.org/blog/the_problems_of_open_source.htm

[15] http://www.phoronix.com/scan.php?page=article&item=sprewell_licensing