Benefits and Challenges of Integrating Open-Source Software Into the Software Development Process

Tyler Conger
College of Engineering
The University of Alabama
Tuscaloosa, AL
tconger1@crimson.ua.edu

***Abstract*—The concept and market for Open-Source Software (OSS) solutions has grown in enormity and become widely adopted in many various sectors of the industry since it's early days. With a promise of cost-savings, increased efficiency, and more flexibility OSS has become very popular with developers and managers alike. As OSS has grown in popularity it is important to understand at what key points it is able to improve upon the software development lifecycle, as it both allows a workflow to be increased tremendously but can also pose some difficulties to unknowing individuals, without proper implementation and awareness. This paper seeks to identify some of those benefits and challenges that occur when integrating OSS. This paper will first explore the potential challenges or downsides of integrating OSS into your project. Some of the predicted downsides include potential quality and security issues, a limitation on perspectives involved in creation of OSS, the fact little to no long-term support may exist, and potential legal and licensing challenges that may occur on the business end. However, while there are downsides, there are also benefits to integrating OSS into your workflow which are equally important to be knowledgeable and aware of. Some of these benefits that will be discussed include the speed at which innovation and development are able to take place, the community support available to these OSS packages and libraries, the general reliability of mass testing and use, and the transparency offered through using OSS. Investigating the potential upsides and downsides that come with using OSS in large or small-scale software projects, understanding that with every benefit there is a hidden deterrent or challenge that exists within.**

Keywords—Open-Source Software (OSS), Free and Open-source Software (FOSS), Software Component, Software development life cycle

I. Introduction

Open-Source Software (OSS) has become increasingly widely used within the software development life cycle in a multitude of sectors of the industry. Allowing for community driven solutions to various problems, both small and large, and allowing for code to be cross-checked and validated by anyone helping to promote transparency within the codebases. This ability to collaborate and work together has brought forth a great wave of innovation, has helped to reduce costs through the reuse of software, and has allowed for increased flexibility and the ease to change between different libraries, frameworks, and codebases. As this methodology allows developers to just take each individual component that is necessary for a project and use it by itself, this style of component-based development has become more commonplace. With the widespread use has also come increased reliability as applications are rigorously tested and verified, following best software development practices. Allowing for the development of in-house components working in tandem with open-source components. However, OSS hasn't been all good, with the upsides have come with some increased downsides as well, and these are equally as important to understand before integrating them into your software development pipeline.

As OSS has continued to grow, it has lost some of its initial charm, in some ways becoming more difficult and overbearing than would be expected. As projects have grown in complexity, new contributors have continually had increasing difficulty with social barriers dissuading some from contributing. Also, as projects have continued to grow, so too have the security concerns. As these code bases may not be entirely vetted or understood by an in-house resource, there are linked security concerns that something malicious may exist allowing for a great deal of potential damage to be dealt. These security concerns and vulnerabilities take time to understand and analyze causing an increased vulnerability analysis time associated with using these open-source libraries, with these security vulnerabilities come business concerns.

Through the process of reviewing others literature and drawing upon real world experiences and practices within the industry, an understanding and analysis of the various benefits and challenges of using these open-source software libraries. It is important to understand and be aware of all facets, including the ones that someone may not think of immediately, such as some of the associated downsides such as the aforementioned social disparities that exist within OSS, including some individuals feeling overwhelmed and unable to contribute. Understanding these complexities, both positive and negative, will allow for a more full and complete understanding of when and how to use open-source software libraries and how to integrate them into the development process.

II. Associated Challenges

While there is a plethora of upsides to integrating OSS into your software development process, there are also plenty of downsides associated with OSS. It is extremely important to understand and be aware of these first and foremost when considering integrating open-source software into your development pipeline. This section will delve into just a few of the potential associated downsides with using open-source software including ideas like the potential quality and security issues that can be associated with using OSS, the potential inequities that are inherently involved in the creation of OSS, how long term and end of life support for some libraries and packages can be more difficult with OSS, and potential legal challenges that are associated with licensing and using OSS within a larger software ecosystem. All of these various challenges that are associated with integrating OSS will be unpacked and understood at a greater length. While these are only a few of the specific challenges

that are faced when using OSS, these are some of the most prevalent and important to be aware of and understand when in comes to integrating OSS.

A. *Quality and Security*

As it is open-source software's nature to have a wide variety of contributors and editors this can lead to quality and security concerns. It is generally unknown, to the user of the library, who these individuals' making changes and edits to the piece of OSS may be, and as such that raises several security and quality concerns, if these potential changes could be malicious. Even while most OSS has a generally accepted release and screening process, it is still possible, especially on a smaller less established project, that something malicious may occur or a damaging bug may be introduced into the code, unknowingly. This can be worrisome to a business or other who is using an OSS project, especially if that OSS code is added to a project with security or reliability constraints. Unless developers are double checking the update logs and each individual change that occurs it can be difficult to track the number of changes that may be taking place between updates, as well as what information or pieces of code actually are changing each time. This is especially true in OSS projects that are lacking in detailed thorough documentation.

One major issue is the lack of documentation associated with these codebases. While some projects may have a great deal of documentation, programmers who add to OSS libraries typically spend time programming and not writing documentation. Because of this, it can be difficult to implement, understand, and interpret what separate functionality does or may be used for [1]. This lack of complete understanding may also introduce security challenges where the software is used incorrectly within an application. Documentation may be much less robust and clear as compared to a closed-source software solution, especially one developed internally. This can cause a great deal of challenge when a system your piece of software relies on is unclear or vague in its documentation, even if you have access to the code, it can still be difficult to parse out and understand what features may be useful to you and how exactly they work. This quality of documentation is not the only issue to consider, there is also a large quantity of various OSS projects to wade through in order to find the best one for your use case.

With each project having different needs and requirements using varied OSS libraries may be necessary, and with there being 1,000s of different options available it can be difficult to know what OSS library is best in a certain situation. Because each project or person may need or want to have their own features there have been a large number of different options created. As such it can be difficult to wade through all of the various projects to find the actual quality ones that are useful [1]. As is reported by GitHub over 30 million different open-source projects exist on within their platform, even with the ability to sort, being paralyzed by the sheer number of different choices is a very real issue. As well as the included time cost spent understanding each one and the expertise required to do so. As such understanding the software and those who developed it can help to alleviate concerns of security and quality that may exist when choosing a library or project to use. These issues can be better alleviated through community support, better documentation, and easily readable and understandable code bases.

Also in large applications, and various web-based applications, like a modern day React website, it is common to use a huge number of publicly available libraries for various different functionality, with even publicly available libraries having numerous dependencies. Oftentimes each of these libraries will also call upon dependencies greatly increasing the number of OSS libraries used. This large number of interconnected libraries creates a huge number of places that must be checked and verified against vulnerabilities. As such tracking and following all libraries and all potential vulnerabilities that may occur is extremely important and can be difficult and costly to track for organizations. While there are tools available to track potential vulnerabilities, adding these tools to your project can come at a monetary cost. Both the security and quality of an open-source software library are closely linked, and as a huge variety of them are used in a single project it can be difficult to track the information about each individual project that is used or imported within a project.

B. *Inequity in OSS*

While on a first glance it may seem that open-source software should be completely fair and open to all, there is a bit more nuance to be understood with this concept. There are barriers to entry that deter and limit some individuals from being able to sit at the table and add to open-source software. Some individuals who want to add to OSS feel overwhelmed or inexperienced when looking at the large complex libraries and packages that exist and can be scared off from participating in OSS based on this feeling. Not knowing where to start or how to add to a project deters many of these individuals from ever contributing to a project. This only hurts everyone involved as it removes a potential developer from working on OSS and also any new ideas or perspectives, they may have brought with them to help improve the software [6]. This problem is also exasperated in regard to minorities such as women in computing. Where women in open-source project contributions represent a small number comparative to that of their male counterparts. Women represent about 10% of open-source software contributors, a small fraction comparative to men [3]. This huge discrepancy showcases one of the major issues, a majority of OSS comes from the same group of people, limiting potential outside perspectives. As a business or developer when deciding to use an OSS project this limiting perspective can showcase how the OSS may be jaded towards one group or another and not encompass everyone involved in using it.

As OSS grows it is important that the variety of people involved grows too, in order to equally benefit each person using that software. We see that this varied perspective has grown helping to improve and move forward OSS. As a varied number of perspectives and opinions will help to improve and make all OSS better for each party involved with it.

C. *Longterm Support*

With any volunteer style software, like open-source software often is, there will be associated challenges. Because each individual group or person only adds and

creates what they themselves need, or see a need for, therefore it is not often times that a singular person supports a piece of OSS indefinitely. Because long term support is not always guaranteed, especially on smaller more specific projects, there can be only a handful of developers pushing out changes and updates, and when these developers must move on the future of their OSS libraries can be unclear. This causes these software solutions to have a potentially shorter lifetime.

This can be an issue when using these libraries within larger products as it may later down the line shift the duties of updates and bug fixes. This type of issue can be worrisome, and thus widely used and supported libraries see more usage rather than more nuanced counterparts. It is also worth noting that any custom modifications or changes must be made by the company, and often times are not released as OSS themselves [1]. This can exasperate the problem as even if the necessary updates or fixes may have been made by another company if it was never released as OSS, it is not helpful to the broader software engineering community and thus must be done again on a voluntary basis. Also, any of these individual changes must be maintained and upkept, which can create a prolonged cost for the business. Often times these kinds of changes are prompted to fit a certain software component within a broader business system, and modifications are needed on the original open-source software in order to make it work with desired functionality, another penalty of a more component driven methodology rather than building software as a single entity [1]. There is also a general feeling of uncertainty regarding the future of various software libraries.

Within OSS as libraries change and progress over time, they may lose some amount of backwards compatibility damaging features you may rely on causing potential issues down the line. This potential for changes can cause worry among some who feel the OSS may change. And these issues down the line could create a great deal of headache adding extra cost and manpower needed to address and fix any potential issues that occur. Whereas with closed-source software this is not nearly as much of an issue as all changes are maintained internally and any damaging changes can be made gracefully allowing everyone appropriate time to react and make any necessary changes or fixes. While this practice is generally accepted in OSS, it is still possible that a change is necessary that would remove some backwards compatibility. While all of the external issues with OSS are important to understand it is also important to understand the valid business concerns that may be associated with the use of OSS in various software projects.

D. Legal Challenges

Because OSS is public the various types of licensing associated with the software can create some concerns and issues among the business, and navigating and understanding these licensing issues can cost time and money. While most OSS libraries and projects use many of the same standard licenses, the MIT and Apache 2.0 licenses [4]. It can still be difficult to navigate the various licenses that do exist especially when using the software for commercial applications. This can cause confusion or potentially legal trouble for failure to understand and navigate the licesing process properly. These licesning issues become exasperated when combining outside OSS with proprietary codebases, mixing these two can cause more potential issues with incompatible licensses. As such it is important organizations are clear on what comes from where and give credit where it is due in line with the associated licenes. As there are a wide variety of liceneses it can be confusing and difficult to keep track of the various meanings and requirements of each license. We can see the some of the various liceneses along with how widely used they are in figure 1 below. The figure showcases how two license styles dominate most projects, but even these two only represent about two-thirds of all OSS projects. As such we can see that there exsists a large variety of liceneses that must be contended with which can cause confusion and various issues. This large variety of licesnes is especially true on smaller OSS projects or ones that require a better understanding to integrate into the development process.

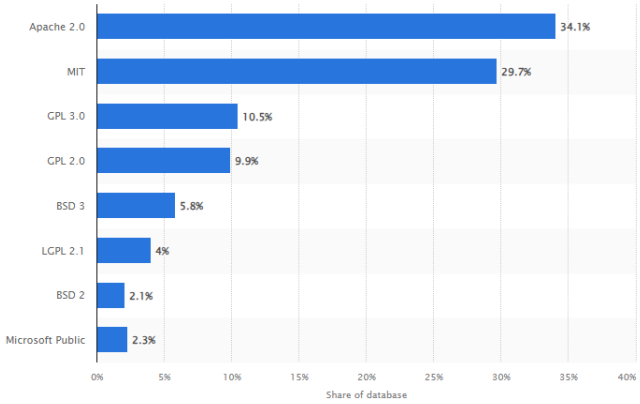

Figure 1. Most popular open source licenses used overall in 2021 according to statista [4].

Another potential legal issue that may occur is that related to intellectual property rights. If code in an open-source package has been illegally used or obtained it may create even more issues down the line as it is reused [1]. Even if the company avoids issue, if the package or library is suddenly taken down or removed, any dependent applications would have serious code breaking issues and need to find alternatives very quickly, again bringing in to question potential reliability issues associated with OSS. This could cause a large amount of development efffort needed to fix an issue like this creating additional cost associated with using OSS.

E. Conclusion

The use of OSS is one that is becoming increasingly more common with the advent of more and more modularized applications that rely on a larger amount of library and package resources written and created by the community. OSS has worked to change the way many software applications are created and architected, becoming more and more ingrained. Because of its widespread use, and the ballooning number of OSS projects, it is important to understand the potential associated negatives or challenges that can come up when using OSS within your own project. Navigating and understanding these potential issues can represent a big piece of how easy it is to integrate pieces of OSS in your project. While only a few of the potential drawbacks were covered here we can see the results of a survey of companies on the drawbacks that exist with the usage of OSS from Morgan and Finnegan. Companies stated the following as drawbacks of OSS; lack of support including compatibility issues, lack of ownership

and expertise, others having access to the source code, no one organization to market OSS, higher investment for training required, and difficult to find the right staff with correct competencies [5]. These issues along with the discussed all work to contribute to the hesitancy in that businesses experience in adding OSS to their projects. While these were drawbacks expressed by the business side of things developers exprese drawbacks including poor documentation, less functionality, lack of a development roadmap, confusing builds, and lack of expertise in certain fields [5]. All of these issues both on the business side and on the development side contribute to potential downsides associated with using OSS, and are only compounded as the number of OSS libraries or packages increases with the scale of a project.

Overall, while there are some downsides that are associated with utilizing open-source software, there are also upsides associated with it. Now that we have covered a few of the potential challenges and understand some potential areas for concern that are associated with OSS, we will now cover some of the associated benefits to using OSS.

III. Associated Benefits

While there are downsides to using open-source software, there are a huge number of benefits associated with the integration of OSS, but with anything it is important to be knowledgeable and understand both sides of the process in order to make a fully informed decision. Now we will cover some of the associated benefits that come with using OSS. Just a few of the potential benefits include, the reduced cost associated, the large communities behind these libraries and packages, the reliability of the software due to mass testing, and the ability to fully and complete understanding of the code base. All these features, and more create a strong case for using OSS within a software project.

A. Innovation and Deployment Speed

First and foremost, using OSS can save time and money when implemented properly into the development process. The ability to take already existing code, and use it to solve a problem, greatly reducing the amount of work required is a huge factor in the use of OSS.

The ability to use and reuse outside libraries and OSS packages allows for quick and rapid deployment. Because a developer does not have to spend a great deal of time spent reinventing the wheel, they are more able to spend time integrating different libraries and working on quality solutions elsewhere in the project. As OSS in its nature has a community spread across the globe with varied backgrounds and histories, they are able to work together to solve problems and find solutions; fixing bugs in such a way that is done quickly and without issue. And because many modern OSS projects have so many quality control gates, following best software development practices, such as allowing everyone to review new code as it is added, this means that OSS is more agile and able to adapt to changing needs of the community extremely rapidly and with ease. The broad spectrum of OSS and the nature of it being publicly available also help to foster innovating within the sector, as anyone is able to see and understand code that is publicly available.

Also, the OSS projects themselves help to foster innovation within the industry, as it allows others to learn from each other and other projects. The ability to see and understand how certain libraries or packages work. Because OSS often follows the CICD model of continuous integration continuous deployment there are constantly changes and adjustments that being made to the software that work to continually provide improvements overtime. This means that OSS will only continue to improve and expand, getting better with each iteration. The iterative process of development for OSS with the community providing new input means that new innovative features are constantly being added and tweaked, allowing for improved more useful software overtime. This can be a great benefit when integrating OSS into your project as it allows for improvements with age as the software is upkept and continually worked on. Also, because OSS is open to the public that means that new achievements and findings are shared lifting all projects up. This allows for small improvements to be continually made on projects in an iterative manner. The ability to collaborate and work together through problems brings a unique wide range of opinions and experiences to the table, changing the way problems are looked at and approached. The great deal of experienced developers that work on OSS projects helps to drive innovation and create and environment where best software practices are followed and respected. These innovative ways allow for increased quality overall and updates and fixes are made quickly to meet the needs of the community.

B. Community Support

The large communities that gather around OSS help to contribute to one of the benefits of using it, community support. As there are a number of individuals who develop a strong expert knowledge on the libraries they work on, and thus the community forums are often able to answer questions that one may have when implementing OSS. This community support network allows for more help on a voluntary basis [7]. On many of these community message boards as well, previously asked questions still exist allowing for a newer user experiencing the same or a similar problem to find solutions simply quickly and easily by searching previously asked questions. This style of message board community interaction is very popular within the OSS community.

Even if the community message boards or mailing list does not immediately know the answer the ability to draw upon a large knowledge pool is extremely helpful. One may be able to ask a unique question to their situation and even if it has not been done before, someone in the community may be able to provide a new or unique idea and approach to the problem. This allows the ability to draw upon a very large knowledge base when working with implementing OSS into your project. These communities can help individuals grow and learn as they progress in using OSS and help more experienced developers get answers to questions that may be extremely nuanced and difficult.

The use of the OSS community can also help to drive changes and improvements to the software. If a large number of individuals are requesting specific features or changes it may steer the direction of the developers into

implementing these changes and modifications. This influence that is held by the community to make changes helps to steer the development to the most useful features to the most people. Similarly, if an unthought of idea is requested this may be the catalyst for development on that previously unrequested piece.

C. Reliability and Tester Base

Open-source software poses a feature that closed-source software does not, it's wide usage. Because OSS is widely used any potential issues are discovered quickly and able to be repaired. This helps to improve the overall reliability of the software. The large number of developers using and interacting with OSS on a daily basis are continually expanding and pushing the boundaries through testing the software. This rigorous testing is constantly identifying issues and reporting them to the community [5]. Allowing such a vast tester base is a way to continually provide improved OSS over time as they are rigorously tested. This testing by the community members helps to remove errors, inefficiencies, and inconsistences that may exist within the code base. These issues are able to be quickly rectified and addressed by the community, again proving the reliability and strength of the community around OSS. This peer review is a large factor that allows OSS to be so successful, even without large teams working on it, as would be the case with close-source software, OSS is able to thrive because of the intensive peer review process and community around each OSS project that exists. It is also worth noting that as OSS projects do not exist in a singular place, they are more resilient to experiencing downtime than would be possible or expected with proprietary software run by a singular organization or business. This resilience and ability to be heavily scrutinized helps to lend strength and reliability that exists within OSS. As we have discussed, the public nature, or transparency, of OSS helps to lend to the benefit of OSS in reliability.

D. Transparency

While using OSS does have security disadvantages, the ability for the community to cross-check and verify code also helps to provide a greater level of security as well as quickly find and squash any bugs that may exist. In widely used libraries, security may be higher as a vast amount of people are looking over the codebase and verifying it, as well any bugs that are found are reported much more quickly. Also, because such a large user community exists, they are able to provide fixes and updates to these bugs more quickly than a small team may be able to do so. This helps to reduce the amount of time that a potential exposure may have occurred, minimizing this time is key in producing a safe and usable application with OSS. The fact that OSS is open and free also helps to provide a level of comfort regarding potential security vulnerabilities. Because OSS is transparent, security researchers, developers, and users are all able to read through and scrutinize the code finding any potential issues quickly. The fact that OSS is transparent helps to provide trust, as anyone is able to verify the contents of what may or may not exist within the codebase. This transparency also helps to understand what practices are used and if best practices are being followed or not. This level of scrutiny allows researchers to flag potentially weak OSS solutions that may need improvement. Overall, the ability for a multitude of individuals to read, understand, run, and test open-source projects allows for a great deal of transparency that can lead to quickly solving problems and squashing any bugs that may exist within the software, helping to improve and provide a better result for all individuals using it.

E. Conclusion

As we have seen, integrating OSS into your software development process does have downsides, but it also has a great deal of upsides involved which is why it has become increasingly commonplace and widely adopted to do so. We have discussed just a few of the potential benefits of using OSS. When surveyed businesses expressed potential benefits as; lower cost including licensing and bug fixing, general flexibility with the licenses, escapes the feeling of being tethered to a single vendor, increases collaboration among teams, increased innovation through access to the source code, increased business functionality by keeping teams smaller and more agile, and community standards that software is held to when available for public scrutiny [5]. Developers cited some similar benefits for using integrating OSS in their development process including, more reliable software, increased security due to source code availability, performance improvements in capacity and speed, large developer test base keeping OSS up-to-date, flexible use allowing customization of software, compatibility with various different systems, and harmonization allowing for easily scalable features and practices [5].

All of these reasons showcase the various benefits associated with integrating OSS into the development pipeline. While there are drawbacks, the benefits are many and lend credibility to the continued development and usage of OSS. It is important to understand both the benefits and the drawbacks when assessing the potential use of an OSS library or package.

IV. Conclusion

While open-source software may at first glance be a scary idea, publishing all your code and using code majorly built by people all around the world that you have never met or been able to certify, it is important to understand the advantages that come with OSS, and how useful it can be to integrate it into your software project. As the ability to innovate and deploy at a rapid pace through usage of previously created libraries and packages. The ability to get real-time support for past and new issues through the community in a forum or mail type setting can be a huge advantage especially in getting advice from a more experienced developer. The reliability of OSS, as it is constantly being tested, updated, and fixed allows for strong software projects, also with decentralized servers downtime is minimal on OSS projects. Finally, the transparency, the ability to actually read understand and interpret the code that you are dealing with allows for a new level of understanding and tinkering that may not be present in a proprietary software solution. As discussed, there are downsides to these though, but overall OSS is a strong solution to integrate within your own project and can help to move projects forward quickly and efficiently.

While this paper simply covered a review of past literature and techniques, it is important to understand what a future piece of research could look like. In the future an analysis between two similar businesses, one that used OSS and one that relies on close-source software would be extremely interesting. This would be especially interesting if we could monitor the activities and success of these two hypothetical businesses over time in an effort to understand what the potential upsides and downsides each faces as they continue with development of a product.

Overall, the use of OSS is a game changer in software development and can help to improve workflow and process. Integrating OSS into a project can have benefits and potential challenges faced for the business side as well as the development side.

V. References

[1] K.-J. Stol and M. Ali Babar, "Challenges in Using Open Source Software," *Association for Computing Machinery*, p. 6, 2010.

[2] T. Vale, I. Crnkovic, E. S. d. Almeida, P. A. d. M. S. Neto, Y. C. Cavalcanti and S. R. d. L. Meira, "Twenty-eight years of component-based software engineering," *ScienceDirect*, pp. 128-148, 2016.

[3] B. TRINKENREICH, I. S. WIESE, A. SARMA, M. A. GEROSA and I. STEINMACHER, "Women's Participation in Open Source Software: A Survey of the Literature," *ACM Transactions on Software Engineering and Methodology*, p. 36, 2022.

[4] L. S. Vailshery, "Most popular open source licenses worldwide in 2021," statista, 2024.

[5] L. Morgan and P. Finnegan, "Benefits and Drawbacks of Open Source Software: An Exploratory Study of Secondary Software Firms," *IFIP International Conference on Open Source Systems*, vol. 234, pp. 307-312, 2007.

[6] I. Steinmacher, M. Gerosa, T. U. Conte and D. F. Redmiles, "Overcoming Social Barriers when Contributing to Open," *Computer Supported Cooperative Work (CSCW)*, pp. 1-44, 2018.

[7] M. A. Khan and F. UrRehman, "Free and Open Source Software: Evolution, Benefits and Characteristics," *International Journal of Emerging Trends & Technology in Computer Science (I JETTCS)*, vol. 1, no. 3, 2012.