Name : Rachel Anastasia Wijaya

Student ID : 2502009646

Course Code - Class : COMP6798001 - L1BC

Taking on the role of the arbitrator between the open-source proponent and proprietary developer, I will be presenting my findings for the case of proprietary software versus open-source software. As the arbitrator, I am tasked with the responsibility of facilitating negotiations and dialogue between the two parties to reach an understanding and help resolve their conflicts.

As with most issues, there are always two sides of every coin. No phenomenon or thing can be purely ‘good’ or purely ‘bad’. In the sense that no matter how beneficial something can be, it ultimately will also have some drawbacks. There is no perfect and flawless solution to anything. This inherent fact is why most solutions tread in both sides’ waters, trying to reach a compromise that is mutually accepted.

With that thought in mind, I will start this discussion with the results of my research concerning types of software in general. According to snyk.io, most software falls under one of two categories, proprietary software, and OSS (Open-Source Software). If a software falls under no license, it is also divided into two categories, public domain software, and private unlicensed software. In this case study, we will be focusing on the former two.

Proprietary software, commercial software, or closed-source software is defined as software that is legally owned by the person who created it. The company that controls the product’s right normally does not reveal the source code to the public, and only those who have paid a specific license key are allowed to use it. Examples of proprietary software include Microsoft Windows, macOS, Adobe Photoshop, Skype, and many more.

OSS, on the other hand, is defined as software that is free and available for the public to access. Creators of open-source software publish their code on sites like GitHub and allow others to use and modify it. As a result, communities often form around OSS, where developers work together to further develop and improve its capabilities. Some examples of OSS include Linux, Mozilla, and Open Broadcaster Software (OBS).

Next, I will be discussing the benefits (pros) and detriments (cons) for both types of software. There are two points of views that will be brought up in this section, as a consumer, and as a developer.

From the perspective of a consumer or buyer of a product, it is easy to assume that OSS is the more convenient and accessible option. Yet, it is not as clear cut as that. In fact, there are some things to consider, like quality, longevity, and the support provided by the product’s provider.

While open-source software do tout benefits like:

* Being free to try before you buy’
* Free support in the form of online forums and communities
* Having generally fewer bugs and faster fixes
* Better security and safety
* Avoids vendor lock-in, or a dependance on a single entity for continued maintenance, longevity, and support

It also is weighed down by some drawbacks, such as:

* Reduced competitive advantage
* Minimal support leverage or quality
* Less usability
* Increased business risk

It might be good for a consumer to try an open-source software easily and freely, but as an OSS developer, one might have to consider the cost and expenditure needed to maintain and care for the product. Considering that most open-source software is largely overseen and developed by smaller companies and groups, it does bring a level of uncertainty when trying to use an open-source software.

Another drawback for open-source software would be the lack of safety in terms of copyright, plagiarism, trademarks, and patents. Being open-source, OSS is open and free for anyone to contribute, fork, or use, except for Commercial Open-Source Software (COSS), in which the copyright, patents, and trademarks are fully owned by a single entity. This lack of protection for the software can result in issues such as the recent Streamlabs, Lightstream, and OBS controversy.

Graphical user interface, website

Description automatically generatedOBS is a free, open-source software that helps in recording and streaming. Streamlabs on the other hand, is built off OBS, and at one point was named Streamlabs OBS before changing to Streamlabs shortly following the controversy. Lightstream is a similar software that was released before Streamlabs.

Figure 1: Comparison of Lightstream (left) and Streamlabs (right) website

An article by Polygon writer Diaz (2021) states that the controversy started when the Lighstream twitter account tweeted out a post comparing the suspiciously similar, down to user reviews, pictures, and layout of the Streamlab website layout to the Lighstream website. On top of this blatant plagiarism, the OBS twitter account also chimed in with a post claiming that Streamlabs had reached out to the OBS creators about using the OBS name. Even though the OBS team was against this, Streamlabs still went ahead and filed a trademark for the OBS name.

Through this controversy, it is evident that an open-source model comes with its fair share of risks and threats, especially regarding intellectual property and copyright. Where consumers can be misled to think that a product is associated with another different product, just because of the gray area it is in in terms of name, trademark, and copyright.

Now for proprietary software. Proprietary software, to a consumer, can be perceived as a sort of investment that they expect to gain from. Most consumers buy proprietary software with the expectation that they will be provided a top tier and best of the best experience. This perception comes from the fact that if you check the data, most proprietary software is owned and produced by larger corporate entities, for example Microsoft.

As such, proprietary software has benefits, such as:

* More usability
* Product stability
* Full rights and ownership, regarding patents, copyright, trademark, and intellectual property
* A more tailored and higher quality support team for maintenance and sustainability

With those pros, also comes some cons, including:

* Customer dependency towards the product provider
* Lack of software transparency

In terms of the consumer’s point of view, proprietary software comes with the added benefit of often coming from a reliable and well-known company, like Microsoft or Apple. Brand awareness and the significant market-share they have make proprietary software appealing for consumers, but at the same time, those bonuses come with a sometimes-hefty price tag. From a developer’s perspective, in the long run, adopting a proprietary software model will both be a big investment, but if it manages to take off, it can become a staple in the industry and be profitable.

Taking the role of an arbitrator means that I will be trying to find the middle ground between these two concepts, while remembering to put aside personal bias to uphold a fair and just analysis of the issue. Though open-source software is shown to be gaining more traction nowadays, with more consumers ranking it high when surveyed. But also, because most consumers are under the assumption open-source software is better than their proprietary counterparts. Neither business model can be said to be the absolute best and most businesses can be successful with either or utilizing both approaches simultaneously (Optimus Information, n.d.). It mostly boils down to your own principles, philosophies, and viewpoints.