Perplex Musical Cube

(Technical Paper)

Linux Actor-Network Analysis

(STS Paper)

A Thesis Prospectus Submitted to the

Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements of the Degree

Bachelor of Science, School of Engineering

Talis Basham

Spring, 2020

Technical Project Team Members

Bryan Rombach

Talis Basham

Shirley Wang

Connor Park

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature _____ Date _____

Talis Basham

Approved _____ Date _____

Harry Powell, Department of Electrical Engineering

Approved _____ Date _____

Kathryn A. Neeley, Associate Professor of STS, Department of Engineering and Society

I. STS Topic Introduction

Free and Open Source Software, hereafter referred to as open source software is software that is distributed under a license that permits it to be freely redistributed. In some schemes known as copyleft it requires the modified version to be redistributed under the original terms. This type of software pervades the modern technological ecosystem. It is easy to take for granted the availability of high quality and modifiable software distributed at no cost to the end user. Although there have been major open source projects from corporate creators in recent years, many of these project were started and initially developed by individuals.

I will use one such high profile piece of software as a case study for examining the growth of a successful open source project, the Linux kernel. What once was the hobby project of a Finnish graduate student has grown into a formidable product with market shaping implications developed by thousands of contributors. Linux can be found in wide array of computers from the pervasive Android cell-phones to the world's most powerful supercomputers. It is apparent that inspecting the process behind this type of explosive growth is key to understanding the unique position that the open source model inhabits in the software ecosystem.

I will begin by exploring the ethics of open source software in general. It provides a unique generation of value for contributors at an individual and organizational level. The success of Linux can be interpreted as the coalescing of individuals and groups around the development and use of the kernel. Towards this end, I will analyze the development of the Linux kernel with the Actor Network Theory framework. This framework describes the formation and operation of a network of groups, individuals, and other actors (Callon 1986).

Linus Torvalds is the original creator of Linux, and he retains final say in conflicts arising in development. He is responsible for determining what additions are accepted into the official version of Linux. This follows a common leadership model in the open source software world: the Benevolent Dictator for Life. This model stands to be contrasted against a more designed by committee method other projects take (Raymond 2000). Important early support came from the existing free software movement and the GNU project. GNU's goal is to provide an open source operating system, consisting of an entire UNIX like set of tools and programs. However, the GNU project's kernel was not ready for use at the time of Linux's development. The tools that the GNU project had already written well complemented the Linux kernel and were readily adopted by it. Linus Torvalds mentioned porting the GNU C Compiler in his post on Usenet first announcing Linux (Torvalds 1991).

As the Linux kernel grew in size and usage, bigger companies began to take interest and contribute to the project. For example, Intel and IBM provided support for using their CPUs. Many companies using Linux in their products choose to submit their modifications to Linux for inclusion in the official version because it is easier to maintain them if they are accepted. Many groups provide their own distributions of Linux bundled with additional software. Some companies provide commercial support for their distribution. Red Hat is one such company that provides professional support for Linux installations. Red Hat employs developers to write not only patches for the Linux Kernel, but also develops user space programs such as SystemD and Pulseaudio. Conversely, there are distributions maintained entirely by volunteers such as Debian.

A Brief History of Linux

In the present day upwards of 80% of changes to the Linux Kernel are written by people

paid by their employer to provide. However, Linux has a reputation for being a community driven effort, and is often presented as a success of non-commercial development. The GNU project dated back to eight years before the announcement of Linux. They had developed much of what would become Linux's userland before its release. Linus Torvalds's original vision was a clone of the operating system MINIX he created as a hobby project to familiarize himself with the i386 architecture(Torvalds 1991). His interest in Linux grew as the problem morphed into that of creating a usable and efficient operating system kernel. This change was driven by its adoption for regular use. The human and organizational actors coalesced around the Linux kernel with the intention of making it more usable for their purposes. I identify this push towards the bettering of Linux for industry use as the problematization central to the actor network which I will explore in my final thesis. Contributions most immediately benefited the contributor, but many of them were widely useful and improved the quality of Linux for other purposes as well. This interest in improving the common ground drove the formation of connections between the aforementioned actors.

The early Linux community created the first Linux distributions which provided simple setup and management tools for Linux systems. These attracted new users by lowering the barrier to entry and greatly benefited the growth of the Linux community. One such early distribution, RedHat, was purchased by a software distributor.

In the nineties Linux experienced significant improvements to stability for information technology industry use. It was freed from its single platform constraints in 1995 when it was ported to DEC Alpha and SPARC, two non-Intel processor architectures common in Unix based workstations and servers of the era. This was followed by other changes which had the result of

making Linux more desirable to information technology applications. In 1998 this desirability was confirmed by IBM's commitment to provide significant funding to Linux development.

Applying Actor Network Theory

I will explore the growth of connections of these actors as the interessement of the network. The diverse groups are connected primarily as a hub around the source of the Linux kernel, although some carry relationships with the other actors. The communication required to develop this took place through the kernel mailing lists. I identify the enrolment of the ANT framework with the actual development of software and activities supplemental to it. The most defining feature of open source software is that it is constantly changing. It is common for older software to be supplanted by a competing better software. In the Linux user space there have been many replacements of old software with the new. This can cause friction between actors. For example the GNU userland can be entirely replaced with other alternatives.

I identify the mobilisation of the network with the act of development. The contributors to the kernel send in their relevant modifications to the kernel development community where they are reviewed through a strict process. However, some patches are developed and distributed outside this structure. The grsecurity patches provide an example of this. They provide a comprehensive set of patches intended to enhance security, which in 2017 they made available only to their paying customers.

Linus takes a very personal involvement in the kernel development, and he continues to have final authority on decisions, making him an inevitable point of passage for the system. He is known to write vitriolic emails in context of technical matters. Around the time grsecurity was being accused of violating the GPL he described their patches as "pure garbage" and lambasted them for not developing their patches within the typical system. He made similarly disparaging comments about Intel's patches following the recent discovery of the "spectre" hardware exploit.

A source of conflict is ideological: the GNU project's free software community proscribes against software that is not compatible with their license, the GPL. This manifests in a stringent requirements for recommendation of Linux distributions. They took offense to the partnership of Canonical and Amazon, and raised awareness of it. However, this dedication is commonly acknowledged to represent only a fringe part of the Linux community.

Ethics

Small open source projects are traditionally driven by gift culture. Under this system people derive status by creating value for others and giving it away. Duplication of effort is frowned upon, so people either create novel projects or coalesce around existing ones (Raymond 2000). However, social status cannot explain the value for large organizations to contribute such as the ones in the network described above. They must either receive benefit from ease of maintenance letting the community at large upkeep their changes, or from the value provided by the software. It is clear that companies such as IBM perceived such benefits when they invested money and effort into Linux (West 2001).

Open source software encourages competition in the marketplace because of its cost. Zero cost sub-licensing guarantees that the software itself need only compete on technical merit. Additionally, services built around open source software face heavier competition because they share common ground in it (Bond 2005).

Copyleft appears to solidify a utilitarian praxis where work submitted through the official kernel development channels benefits the entire network. However, this can be subverted by groups acting unethically. The grsecurity patches brought about accusations of violating the GPL

because customers claimed that they strongly suggested that their customers shouldn't redistribute the patches. Another case is that of "Tivoization" denounced by the GNU project where open source software is used in a product where the software cannot be modified. They claim that this process harms the end user by taking away their freedom, but it does not harm the kernel development community because the terms of the GPL are followed.

Conclusion

The Linux kernel stands as an example of successful open source software development because a structure developed around it that allows it to effectively utilize the various assets at the disposal of the community. It continues to be developed and used heavily because of the network of contributors and users that has grown around it. In the process of writing the final thesis there are a few areas that I would like to take a deeper dive into. I will look for older historical statistics on the contributors to the Linux Kernel. Statistics are readily available for affiliations of current contributors, but further back they do not seem to have been compiled. I'm particularly interested in discovering precisely when the development shifted from predominantly hobbyists to a majority of corporate contributors. I would also consider expanding my scope to examine other open source projects because Linux's licensing seems to have been a major factor to its success. Perhaps it could be best contrasted with GNU Hurd. Hurd was GNU's second attempt to develop a kernel and its development preceded Linux's release by a year. However, to this day it is not in a stable state.

II. Technical Topic Introduction

My Capstone group set out to develop a musical toy product. We decided upon creating a

cube that would light up and play sounds when the sides were pressed. We focused on creating an affordable design using cheap, readily available parts.

We began by conceptualizing what our product would look like. We settled on a minimalist design of a matte translucent cube.The device is mechanically simple. It is housed in a transparent plastic cube with a latching hinged side. The battery pack, circuit board, and speaker are mounted to an internal frame. This frame provides bracing for internally mounted force sensitive resistors which detect flex in the sides. The circuit board houses the microcontroller, audio amplifier, and orientation sensors. The audio subsystem, which I was responsible for, follows the general philosophy of providing a cheap product. It utilizes the hardware of the microcontroller to use one of the simpler digital to analog converter architectures: pulse width modulation.The lighting uses a common serially controlled RGB LED, the WS2813. A single strip is run around the peripheral of the cube zig-zagged into 4 by 4 matrices on each face. The lights glow with varying intensity based on the pressure applied to the side, and the central ones light up more brightly.

My capstone project provided a valuable experience in product development and embedded systems design.There are a few changes we would make if we were to take the product to production. The most likely components to be replaced if we decided to produce this at a larger scale are the orientation sensors and the force sensors for the sides. They are both relatively expensive compared to the other components. The accelerometer for orientation could be replaced with a 6 axis tilt switch and the force sensitive resistors we used are less reliable compared to ordinary switches.

References

Bond, H. S. (2005). What's So Great About Nothing? The GNU General Public License and the Zero- Price-Fixing Problem. *Michigan Law Review*, 104(3), 547 - 571. Callon, M. (1984).

Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of St Brieuc Bay. *Sociological Review*, 196 - 233. Elder-Vass, D. (2015).

The Moral Economy of Digital Gifts. *International Journal of Social Quality*, 5(1), 35 - 50.

How the development process works (n.d.). Retrieved from https://www.kernel.org/doc/html/v4.15/process/2.Process.html.

Lee, G. K., & Cole, R. E. (2003). From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development. *Organization Science*, 14(6), 633 - 649.

Raymond, E. S. (2000, August 24). Homesteading on the Noosphere. Retrieved from http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading/.

Torvalds, L. (1991, August 25). What would you like to see most in minix? Retrieved from comp.os.minix.

West, J., & Dedrick, J. (2001). Open Source Standardization: The Rise of Linux in the Network Era. *Knowledge, Technology & Policy*, 14(2), 88 - 112.