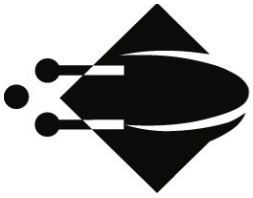
Replacing the X Window System:

Looking at how Free Software Organizations replaces legacy code and how new standards are being created

Vegard Torvund



IT University of Copenhagen

MA European Studies of Science, Society and Technology (ESST)

Centre for Technology, Innovation and Culture

UNIVERSITY OF OSLO

3. November 2014

Student:	Vegard Torvund
E-mail:	vegartor@student.sv.uio.no
1. Semester	University of Oslo
2. Semester	IT University of Copenhagen
Specialization:	Situated Analysis of Global Connections
Supervisor:	Rachel Douglas-Jones
Words:	17666

Table of Contents

Illustration Index

Abstract

Organizations involved in Free Software have pioneered new ways of thinking about intellectual property, collaboration and elaboration. These organizations are bound together in varying degrees, through means of technology, legal circumstances and common goals in an environment that is in a constant cycle of change. This thesis looks at a case where actors within the realm of Free Software are in a process of replacing one standard with a new one. The necessity of gaining consensus on standard ways of enabling interaction between the technologies that they create and maintain is something that is required for these projects in order to achieve stable points of interaction throughout their cycles of development. The thesis investigates a case that carries with it a controversy; the controversy is explored through interviews and online material analysis by drawing on methods and concepts from Science and Technology studies (STS).

First, the thesis explores in which ways the organizations are tied together and what considerations they make when choosing the technologies they use in their solutions. Second, it looks at how the process of creating a new standard have initiated not only a new alternative to the initially proposed standard, but also an underlying debate on whether a standard is necessary on the discussed layer of technology. The findings reveal how software is being used as an argument in discussions about how distributed infrastructures should be designed and how these arguments can be shaped by values. The thesis also serves as an example of how efforts of standardization can be counterproductive in the sense that the end result can turn into a more complex infrastructure than what was first intended.

Key words: Free Software, open source, software innovation, software standardization, reflexive standardization, recursive public

Acknowledges

I would not have been able to write this thesis without the help of certain people. First of all I want to thank my supervisor Rachel Douglas-Jones, who not only quickly recognised what I intended to do with the case, but whose positive attitude and valuable suggestions supported me throughout the writing process. I also want to thank Martin Gräßlin, Clement Lefebvre and Oliver Ries for their forthcoming attitude and for allowing me to make use of their time and thoughts when interviewing them. Also a big thanks to Paul Martin Mauget Birkeli who helped me with necessary cleaning up and polishing of the thesis.

I also want to express my gratitude toward the Free Software community, who continuously surprises me with their great ideas and visions for the future. And lastly to my partner Iselin Grayston, who during these months has supported me with comfort and new perspectives. This would not have been possible without you.

1. Introduction

Have you ever wondered about what is happening when you type a message into your smartphone and send it to a person twenty thousand kilometres away from you? How many computer systems does the message have to visit before it has been relayed to its receiver? What type of systems are making sure that your message is safely transferred to your intended receiver and not someone else? In many cases, these types of questions can not be answered with a great amount of certainty due to the complexity of the infrastructure that enables this type of communication through what we call the internet. The internet is for many people something that is just there for you when you need it; it has become one of our many invisible infrastructures. Yet, some people dedicate their whole lives to the designing, managing and developing of the systems that keep the bits and pieces of our internet together. One group that has played a great role in the creation and development of many of the parts that constitute the internet today is the people and organizations involved in Free Software projects.

Today, these software projects have extended their applicability and are being implemented and used in our smart phones, personal computers, cars, air-plane infotainment systems, ATM's and even four hundred kilometres away from the Earth, inside the International Space Station (GENIVI, 2014;Foundation, 2014;Gartner, 2014) The projects themselves are being driven by various methods of cooperation and elaboration between many organizations, some of which are bound together by common interests, values and goals.

This thesis will take a closer look at some of the organizations behind these projects, and more precisely look at a case where these organizations are within a process of collaboratively changing out a central part of the code which these organizations share and use in their

projects. The same code also binds these organizations together through a common means of association.

1.1 Context

The spawning rate of Free Software projects has during the last 25 years been increasing, and an estimation from 2008 suggests that the total number of Free Software projects have reached the number of 195,000 (English, 2008). If we look at one of the biggest of these projects, the Linux kernel project, the development growth of this project is observed to be "super-linear", meaning a very linear growth in terms of lines of code being written every year (Godfrey & Tu, 2000). Combined together, all these software projects constitute a vast body of recipes and techniques on how to process, store, send, receive and display information. Everyone can observe Free Software projects, and everyone with the sufficient knowledge can participate in its growth and forming. Anyone can make use of the bits and parts of this body in their own projects, usually on the sole premise of redistributing the changes they make back to the body.

This reuse of code, however, does cause dependency situations where some projects become dependent on software that is being maintained by other organisations. If an organisation for some reason stops maintaining their software, the projects depending on this software can either decide to replace this part with a substitutionary part from the Free Software body, create their own replacement, or continue to maintain the software themselves in their own direction, the latter is often referred to as software forking[1]. Software forking happens all the time and is an central part of the evolution of Free Software (Scacchi, 2004). End user software such as video players, text processing tools and similar applications are being forked or replaced with newer software all the time. This process creates diversity that enables competi-

[1] Forking – In software engineering used when a developer copies the source code from a project and start its own independent project based on this code.

tion between the software producers and a choice for the end users. To give an example, the development of the office suite Open Office, declined around 2008 (at that point maintained by the Sun corporation). As a result, the office suite was forked and re-released with the name LibreOffice with new maintainers preventing the code from becoming obsolete for future users and use cases. For infrastructural software such as protocols or core infrastructure that is shared by several Free Software actors, the focus on standardizing is however more stressed (Kelty, 2008). Anthropologist Christopher Kelty tells the story about the communication protocol TCP/IP and how this protocol became a *de facto* standard after a long and almost, as he calls it, "religious war" against its competitor the OSI standard (Kelty, 2008 p. 166). Similar cases can be found if looking at the so called UNIX wars in the late 1980s and early 1990s where software engineers tried to agree on standards for the closed, but still semi-opened operating-system called UNIX. These types of standardization processes can be studied both with a strong focus on the technologies that are being changed or agreed upon, but they can also be analysed through a more socio-technical oriented approach (Larkin, 2013). Science and Technology Studies (STS) provide a wide set of tool kits that can be used in order to do the latter. What I want to do in this thesis is to investigate a controversy similar to the TCP/IP debate and by drawing on concepts from STS, try to increase our understanding about how actors involved in such controversies decide, prioritize and act with the overall goal of contributing to our understanding of how software standards are being created and replaced. The reasons I find a standard controversy particularly appealing is because this type of process reveals some kind of information about what is happening when software evolves.

1.2 The case of X window system, Wayland and Mir

One of the oldest projects in the collection of Free Software projects is the X Window System. I will refer to the X Window System, or just X, as an umbrella term for the reference implementation X.org and the protocol X11. The project ordinates from Massachusetts Institute of Technology (MIT) back in 1984 and has been maintained by the X.Org Foundation since 2004, an educational non-profit corporation that serves the purpose of maintaining an open source implementation of the X Window System (X.org, 2014). The core functionality of X is to draw graphics on the user's screen as well as mediating user peripherals such as mouse and keyboards. X is being used by almost all the major Linux operating systems. In Figure 1, you can see one of the first implementations of the X display server next to a modern implementation in figure 2.

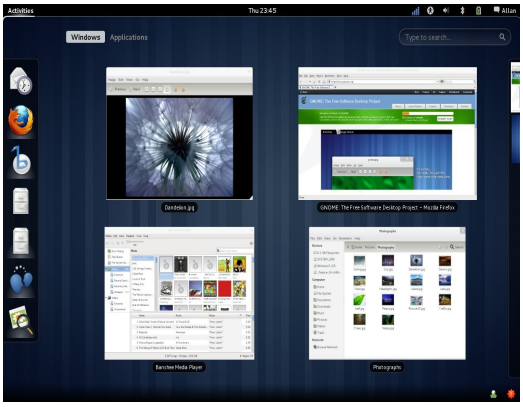


Illustration 2: Modern implementation (2012) of X Window System using Gnome3 desktop environment

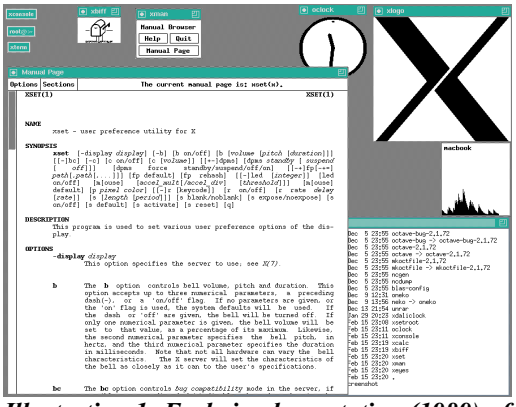


Illustration 1: Early implementation (1989) of X Window System using twm window manager

Due to its long lifetime X has grown to be a very comprehensive project, it was also written in a time when computing memory and storage were more limited than they are today. As a result of this, limitations to its code base make it hard for operating system providers and others to utilize high end performance from X, especially on the new mobile phone and tablet

form factors[2], which would be better by off using a more lightweight[3] alternative to X (Shuttleworth, 2010). As a result of the growing perception that X has become more or less a legacy[4] system, the majority of the actors who are dependent on X have indicated that they will put their effort into a new project called Wayland created by Kristian Høgsberg, serving the purpose of a lightweight alternative to X (Høgsberg, 2008;Jackson, 2010).

In March 2013, however, the company behind the Linux operating system called Ubuntu announced that they wanted to go in their own direction and as a result created a second alternative display server called Mir (Larabel, 2013). This decision created a huge fuss amongst the actors supporting the first project, Wayland. The company behind Mir was then criticized for creating unnecessary fragmentation and acting contrary to the "open source ethos" (Dalziel, n.d.). Some have also urged developers to not accept patches that would enable support for the Mir project (Graesslin, 2014).

---

2 Form factor – Used in describing different physical dimensions of a computing device.

3 Lightweight – Used in emphasizing that some code are more efficient than other code, often written with less lines or in a more efficient or intelligent way.

4 Legacy system – An old method, technology or computer program that is outdated but still used.

1.3 Research objective

The case enables an opportunity to look at what is happening when an infrastructure is in the process of a considerable change. The present controversy between the two different groups of actors, which I just described, is amplifying what is at stake for each actor when they need to decide how to manage this change. My goal is to capture some of the considerations while they are being discussed in order to identify the motivations behind the decisions that are being made. The following research questions will be central in the investigation:

(I) In which ways are the actors in the controversy connected?
(II) What considerations do the involved actors take into account when making decisions about which technologies they are to include in their projects?

The purpose of the first research question is to lay the groundwork for a platform where the case can be further investigated through the second research question. Themes from STS such as studies of infrastructure, standardization, values and politics of artefacts will be shaping how the questions are being answered. The overall goal is to contribute with empirical findings and reflections to our understanding of how Free Software evolve, and also contribute to our knowledge about infrastructures and standardization processes.

1.4 Thesis structure

The thesis will be structured in three parts. First, I will present a brief literature overview of existing research on Free Software together with an introduction of the key concepts that I will draw on when analysing the empirical data. I will start out this chapter by briefly discuss the idea of Free Software and clarify the term which I have chosen to use when writing about the subject. The second part will be covering the methodology and approaches for data collection that was used when I mapped my research objects and planned out the interviews. The two last and most substantial parts will first present the events that lead up to the controversy chronologically; and then the empirical data will be revealed through discussions in relation to the concepts introduced in the literature review. Lastly, I will discuss where the resulting knowledge can be applied and conclude my findings.

1.5 Notes on my own position and background

Since this thesis will be an investigation of a controversy, I find it sensible to write some lines about my own background and position in relation to what I will write about in this thesis.

My interest in Free Software started out when one of my neighbours, who was involved in the Mozilla Foundation told me about this new Internet browser called Firefox, which this foundation released in late 2002. I had been using some similar products in the past but was never aware of how these types of applications came to be or who were behind them. This awareness changed as I read more about the Mozilla Foundation, a non-profit organization that created software. From this point on, I started to discover more and more projects and eventually found myself using several Free Software projects in my daily work. What I found appealing in these was the ability in which I was able to learn how the software worked by in-

specting code and the ability to fix things if something was not right. As the years went by I started to work as an IT consultant and was in this position able to apply much of what I had learned through my previous experimenting into work practice. As a user and maintainer of my own implementations of these various software projects, I gradually became more aware of how they were developed in a collective effort, by subscribing to mailing-lists, engaging in forums and participating in IRC discussions. The incentive for this participation was that I would have a much more control of the projects that I myself used, the control was in form of increased knowledge about the projects, being able to give my feedback and also in order to foresee future features or the lack of them. The subscription to these channels also involved projects that I did not necessarily use, but which I still found interesting because I could see myself using the projects in the future.

This engagement also involves projects that I will be writing about in this thesis. That being said, I am not affiliated with any of the organizations in any way. The field of display servers is also a new topic for me, which I had not spent much time on prior to this thesis. But I find my prior experience very helpful when investigating a case like this since it involves looking in similar types of media channels as well as understanding the terminology and jargon used by the actors, jargon that I will do by best to translate or expand and explain where I find it necessary and relevant for the understanding of the thesis.

1.6 Notes on the relevance of Free Software in our society

In order to comprehend the important role that Free Software plays in our society it is important to be able to think about it broadly and not always just from a one case perspective. In Pinch & Collins two books *The Golem at Large: What you should know about technology* and

The Golem: What you should know about science, we get introduced to a creature called a golem.

A golem is a creature of Jewish mythology. It is a humanoid made by man from clay and water, with incantations and spells. It is powerful, and grows a little more powerful every day. It will follow orders, do your work and protect you from the ever threatening enemy. But it is clumsy and dangerous. A golem is not to be blamed for its mistakes; they are our mistakes. A golem cannot be blamed if it is doing its best. But we must not expect too much. A golem, powerful though it is, is the creature of our art and our craft (T. Pinch & Collins, 1998;Collins & Pinch, 1993).

The golem, used as a metaphor for Science and Technology, can also be helpful when thinking about software. In extending this metaphor we could say that software in many cases represents the wiring of the golem's brain, it constitutes the instructions in which the golem is able to communicate with its masters, as well as to operate its body. Thus, if there is no way for us to review the software that we use when communicating with the golem, how can we then be sure that the golem is following the initial order of its master?

The recent revelations by Edward Joseph Snowden has shed light on some examples of how software can be used in order to harm its users, without the users being aware (Greenwald, 2014). Free Software serves as one of the tools that can help to prevent matters related to the latter, and has indeed been one of the many selling points used by Free Software advocates in the last two decades

> “We now live in a world when the software industry is re-concentrating inside mobile computing devices. The idea that you can be sold a computing device that you can't understand, can't study, can't change, can't fix and therefore can't

> prevent you from spying on you, is the advent of the industry, the GPL[5] and other FOSS[6] institutions were designed to prevent". (Moglen, 2013)

These words by Prof. Eben Moglen spoken out in a plea before the European parliament in June 2013 points out how the Free Software are as relevant in today's society as before. If we are not able to study or understand the brain of our golem, how can we then know if it is not doing us any harm?

The properties of Free Software are however not limited to the transparency of the technology and to technology per se. This thesis will touch on some of the dynamics of how software are being produced and how problems are solved in a collective enterprise, these ways of distributing innovation may also be applied to other domains in addition to the software industry. Industries such as the auto-mobile industry are already trying out modulations of the patent licensing practices that emerged together with Free Software. Here from Elon Musk, founder of Tesla Motors: "Yesterday, there was a wall of Tesla patents in the lobby of our Palo Alto headquarters. That is no longer the case. They have been removed, in the spirit of the open source movement, for the advancement of electric vehicle technology" (Musk, 2014). This trend can also be seen in other research and development industries where a focus on open innovation is gaining attraction (Chesbrough, 2005). This development indicates that this way of e.g., thinking about intellectual property and collaboration could also be adopted by other types of organizations such as our institutions.

---

5 GPL – Genearl Public license, the most used Free Software License
6 FOSS – Free Open Source Software

2. Exploring existing literature and concepts

Much of the research on Free Software consists of best practices and similar research connected directly to the field of Computer Science in the form of technical analyses and white papers which describe technologies. There are however many works where the subject has been explored by other disciplines including Economics, Innovation studies, Law studies and also by ethnographic researchers. I will in this chapter first discuss the term and definition of Free Software, then look briefly at what exists of previous studies on Free software in order to give a sense of where my thesis contributes in relation to these studies. Lastly, I will move on to explain some of the concepts used by STS scholars in studies of infrastructure and standardization which will be applied to my case in the empirical part of the thesis.

2.1 What defines Free Software.

Free Software, Open Source Software, Libre Software, or just the initials FOSS/FLOSS/OSS, are all being used to describe the somewhat similar phenomenon. There is an ongoing discussion on which one to use, and the various uses derives from different ideological branches in the FLOSS spectrum. The Open Source definition introduced by computer programmer Eric S. Raymond is often associated with the more business oriented aspects which focuses on the economic and pragmatic benefits enabled by the practice and were adopted by several companies and individual people during the Dot-com bubble in the late 1990s. The name simply refers to the opening up of a black-box and thus revealing the source (code) of a software program (Kelty, 2008). On the other hand you have Free Software, an older term coined by another programmer named Richard M. Stallman. When using the word “free”, Stallman refers to the matter of liberty and should not to be confused with the word *gratis* which means to “provide something for free.” Stallman highlights the following freedoms of software that can

be categorised as Free Software: Freedom to run, copy, distribute, study, change and improve the software (Stallman, 1996) . These freedoms are protected by one of the many copyleft[7] licenses such as the General Public License (GPL) which is being used by over 50 percent of the Free Software projects around the world (Haller, 2013). The various software licenses vary in their ability to protect these freedoms. An example where source code has been made available but protected by an almost counter-free license can be found by reading the license agreement for the almost 30 years old and successful operating system MS-DOS created and released in 2014 by Microsoft (Gates, 2014). This license prohibits redistribution and use in any other context, and even limit people from citing more than 50 lines of code in written works. The source code itself is still open for everyone to study and can easily be misperceived to be open source.

Due to the broad usage of the term "Open Source" (as exemplified in the case of MS-DOS) I will stick to the Free Software term since the term itself covers the activities pointed out by Stallman, and also because it is the oldest term used to describe the activity. Moreover, because Free software does not only denote the practice of making software source code available to others, the practice of sharing, managing, cooperating and writing licenses are all activities closely connected to the term itself and some of these activities will be touched upon in this thesis (Kelty, 2008).

[7] Copyleft – the practice of using copyright law to protect computer software with various rights, such as using, modifying and distributing.

2.2 Reviewing existing studies on Free Software

As mentioned, previous research on Free Software can be found in various academic disciplines and relevant for my case are those studies that touch on innovation processes. Hippel & Krogh have collected studies from the field of innovation and divided them into three categories: Motivations for contributions; Governance, organizations and innovation process; competitive dynamics (Hippel & Krogh, 2006). This categorisation is helpful in order to recognize where the different themes in my thesis contributes to existing studies.

In the first category which covers studies of individual incentives for contributing to Free Software projects, M. Bergquist & Ljungberg looks at Free Software contributions as gifts in a gift economy where the gifts in the form of code are incentivised by the desire of getting their ideas out in circulation (2001). Others looked at the architecture itself as a factor that generates an incentive for contributors. Modular codebases and codebases with option value[8] create more opportunities for the contributors than codebases that do not (Baldwin & Clark, 2006). Both these studies are interesting because they provide a basis on what might be core incentives and motivations for people and organizations to engage in Free Software projects; motivations that could also explain decisions my actors need to consider.

Another study on motivational factors for contribution looks at the relationship between intrinsic and extrinsic motivations where they found that paid contributions and status motivated contributions lead to higher contribution levels (Roberts, Hann, & Slaughter, 2006). A contradicting study done by Lakhani & Wolf found the opposite, namely that enjoyment-based intrinsic motivation is the biggest driver in contribution (Lakhani & Wolf, 2005). Although these studies are not directly related to what I am investigating it highlights the fact

[8] Option Value - Used to describe the value of maintaining something that might be used in the future.

that people involved in Free Software cannot be looked upon as one group or a group with a certain set of values, as in every socio-cultural setting you will find variations, generalising my actors could therefore potentially weaken my analysis.

In the second category where they look at Governance, organization and innovation processes Mockus, Fielding, & Herbsleb has compared two Free Software projects with several projects developed using traditional commercial methods of software development. They conclude or suggest the usage of a hybrid Free Software/commercial approach to possibly be the most efficient (2002). Yamauchi, Yokozawa, Shinohara, & Isdhida, also finds that the organizational culture within Free Software projects challenges the traditional ways of producing software (2000). Ferraro & O'Mahony also looked at this organizational culture but with a focus on the different adoption of bureaucratic and democratic mechanisms in these organizations and how governance systems hence have evolved through different views on authority (Ferraro & O'Mahony, 2007). Others have looked at the evolutionary patterns of how Free Software grow and evolve over time which I find very interesting since this is one of the aspects which triggered my interest to investigate a controversy like this. (Scacchi, 2004;Godfrey & Tu, 2000). Noteworthy from Scacchis research on software evolution is that he finds that Free Software does not seem to fit any models used when studying closed-source software evolution patterns. This could be because Free Software projects are more dependent on their communities and spread out globally. He writes:

> "There is a growing base of data, evidence and findings from multiple studies of F/OSS systems that indicate F/OSS systems co-evolve with their user-developer communities, so that growth and evolution of each depends on the other. Co-evolution results of this kind are not yet reported for closed source systems, and it is unclear that such results will be found" (Scacchi, 2004 p.22).

He also calls for the creation of new methods to investigate Free Software evolution but emphasizes the need for any method to be focusing on the technological regime as a major element when modelling. A slightly different weighing than what I am intended to in this thesis, which will be to give a great deal of attention to the social elements of the activities.

Some have also looked at how companies outsource important resources to Free Software communities which create symbiotic relationships and increased value but also managerial issues for the companies (Dahlander & Magnusson, 2005). The latter example was found in the category of competitive dynamics and is also relevant for my case since some of my actors are trying to combine commercial interests with cooperation with the community, which could have an effect on the way things are being worked out.

The review at large concludes that free software contributors have pioneered new ways of thinking about intellectual property and about how organizations innovate, while also emphasizing that these ideas could be adopted by economic or other social activities (Hippel & Krogh, 2006).

With a slightly different focus, Christopher Kelty in his book *Two Bits*, tells a story about how the social configurations within Free Software can be perceived, together with a historical perspective of the emergence of Free Software, which includes stories about the standardizing processes of the parts that constitute our internet and the UNIX operating system, the early building blocks of many Free Software projects today. I find both stories relevant in putting my case in a historical context. Not only because the history is interesting but because it shows how software are never built from scratch, but rather evolve from previous work in cycles of change, a central theme in this thesis (Edwards, Jackson, Bowker, & Knobel, 2007). In the book, Kelty also introduces us to the concept of a *recursive public,* a concept which I decided to use in my analysis.

2.3 The Recursive public

To be able to understand why people involved in discussions and debates about the two future alternative display servers, and why they care about it, we need to understand why people involved in Free Software associates with one and another in the first place. In trying to explain this, Kelty uses the concept of a *recursive public* (Kelty, 2008). This concept will serve two purposes for my analysis, the first is to identify the associations my actors are connected through, and the second is to create some sort of socio-technical or actor-networked order that will be useful when I investigate the second research question.

The recursive public is built on the idea of a public sphere where meanings are articulated, distributed and negotiated as a collective body that is constituted and driven by "the public" hence open to everyone (Habermas, 1991). The other term recursive, is obtained from mathematics and computer science and are defined by the Oxford dictionary as "Relating to or involving a program or routine of which a part requires the application of the whole, so that its explicit interpretation requires general many successive executions.". Kelty uses the recursive public concept in explaining the phenomenon of free software for two reasons: first, in order to show that this public involves the activities of creating, maintaining and changing software and networks together with the conventional discourse that follows these activities, the people involved in Free software projects express ideas, but they also express infrastructures through which ideas can be expressed and circulated in new ways, thus do they not only argue about technology, they also argue trough it; second, the recursive "depth" that free software projects usually have, either through technical (e.g protocol and application) or legal abstractions are reflected in the people's ability to see connections between the different layers and draw out implications for the different layers. To give an example of such a layer: a Free Software project may depend on another piece of software which is itself dependent on a particular operat-

ing system or open protocol (Kelty, 2008). If we take a look at the widely used web browser Firefox created by the Mozilla foundation as an example, this application itself is built using libraries and tools maintained by other projects, one of these are the OpenSSL project, which is an independent project used by several other projects (Cox, 2014). The discourse and discussion about which of these dependencies a project should choose to use is, therefore, playing an important role in managing a Free Software project. This discourse can in some cases be politically motivated driven by a certain ideology, goal, or interests, but if we look at the actual discourse, we will find that the discussions are usually motivated by technicalities often using technical arguments to highlight arguments. An example of such argument can be seen in a discussion on the Debian operating system mailing list where those involved discuss the unnecessary code syntax of a particular program (Heen, 2001).

This recursive way of distributing work, but in a slightly different way can also be seen in the context of software designed for enabling people to do collective work (Computer-supported cooperative work) wherein the dimension of articulation work, the cooperating persons needs to split their work into different parts, divide it amongst themselves and, after the work is performed, regenerate it (Schmidt & Liam, 1992;Sarini & Simone, 2002). This delegation of work is explained by Star & Strauss as: “Cooperative work interleaves distributed tasks; articulation work manages the consequences of the distributed nature of work (..) managing articulation work can itself become articulation work, and vice versa, ad infinitum.” (Star & Strauss, 1999 p.10).

Also, In proposing a design theory that tackles dynamic complexity in the design for information infrastructures defined as shared, open, heterogeneous and evolving socio-technical Information Technology capabilities Hanseth & Lyytinen uses the creation of the internet as a reference point in creating different design rules. Information Infrastructures should be de-

composed recursively into separate application, transport and service mechanisms that implement these capabilities as to maintain loose couplings between the connected Information Infrastructures (2010). Information Infrastructures is shaped by both traditional and emerging organizational structures and practices, there is both a moral and technical order which is being expressed in software, hardware and networks. A collective imagination of the proper order in which to arrange these artefacts in supporting different areas of the society is being constantly negotiated and argued both with and through the technology (Lee, Dourish, & Mark, 2006 ; Kelty, 2008).

By reconstructing the layers in which my actors are connected, and by trying to identify what kind of arguments the actors are expressing when creating their technologies, the concept of a recursive public can help answer my first research question.

2.4 Path dependence

My second concept which I will make use of is path dependency. Path dependence within the theme of Information Infrastructures often refers to the "lock-in" effect you get by depending on certain technologies (Edwards et al., 2007). In the context of Free Software "vendor lock-in" is usually used to describe the lack of interoperability and irreversibility when using a proprietary closed-source implementation of an information system. Free Software is by many considered as a way of reducing this type of lock-in (Ven, Vereist, & Mannaert, 2008). But if we apply the concept to Free software projects in isolation, we might also find vendor-lock-in situations, considered that they are built on existing code which is maintained by another organization, which in this case would act similar as a vendor would do (Ghosh, 2003). "Technological change is always path dependent in the sense that it builds on, and takes for granted, what has gone before." (Edwards et al., 2007 p.17) For Free Software projects this might not

be only true in the social dimension; it might also be enforced by the code that binds these different organizations together (Kelty, 2008). Although the code theoretically can be changed, it would still require resources from the organization in the form of labour that writes code.

When economists talk about path dependency they usually look at the efficiency by comparing existing technologies with new ones by measuring cost and labour time (Edwards et al., 2007). Some argue that such measurements are impossible to determine, considered that you would need a widespread implementation of the technology in order to measure all the real-world ramifications (Davis, 1985). Assuming that the reality is somewhere in between these two approaches we could say that it at least can be hard for an organization to decide between two unrealized technologies even by using other measurements. So if you are dependent on a certain path, the irreversibility makes it even harder (but maybe easier for the decision maker) (Edwards et al., 2007).

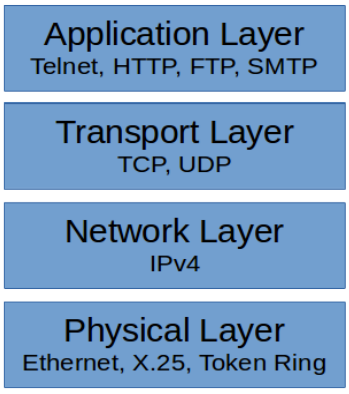
When I am analysing my case, I will take a look at to what extent decisions made by the actors are ruled by previous decisions (what technologies these projects are built upon). Moreover, to what extent my actors are able to steer away from those paths. The concept of path dependence will serve as a way for me to situate myself in the position of the actor, and draw out implications for the actor based on previous or upcoming decisions. By doing this I will be able to review if the decisions creates lock-in situations or not.

2.5 Standardization and reflexive modernization

This case is also about how a set of standardised practices are being replaced by new practices. By using the word practice I speak of two things, first and most obvious the practices in which the organizations exercises when producing their software, but I also speak of the prac-

tice that the software project itself or the code conducts. The practice which the projects Mir, Wayland and the X Window System exercise can in laymen terms be explained as an "instruction for your computer to tell its monitor what colours each of the pixels should be at any time". Since this practice is something that many users of a UNIX like system needs, a common way of doing it has been practised and shared among creators of such systems. It is not per se a *de jure,* or formal standard, imposed by law or standardization bodies as the International Organization for Standardization (ISO), who holds the Open Systems Interconnection model (OSI) where many of the protocols used on the internet are characterized and defined (ISO/IEC, 2014). It is more a case of a *de facto* standardization emerged through market mechanisms and best practices amongst the actors (Hanseth, Monteiro, & Hatling, 1996, p. 411). It also differs slightly from communication protocol standards, such as those found on the Internet protocol suite, see illustration 3. Communication protocols require all actors to speak through the same protocol in order to be connected to the same network at all times while application standards only interact at certain points of releases. If we look at the Internet protocol suite, standards like TCP and IP in the different layers have been stabilized and achieved closure over time through processes of standardization see (Kelty, 2008;Hanseth et al., 1996). The standards are as time goes being more and more irreversible since all actors and intermediaries need to achieve consensus on changes (Callon, 1991). Hanseth et al. has explored the tension between standardization and flexibility in Information Infrastructures, and by flexibility they talk about the ability for the Information System to adapt to changes and not the concept of Interpretive flexibility used in STS. *"With-*

Illustration 3: Internet protocol suite



in computer science, the term flexibility has a different meaning than the term 'interpretative reflexibility' in STS. It denotes either (a) flexibility in allowing further changes or (b) flexibility in pattern of use." (Hanseth et al., 1996).

If an Information Infrastructure is going to continue to live, it also has to continue to change during its lifetime, yet there is a conflict between flexibility and standardizations (Hanseth et al., 1996). An example of such irreversibility or lack of flexibility can be found if looking at the process of routing the internet traffic through the IP protocol version 6, a migration from version 4, a process which has been ongoing since the mid 1990s with a limited deployment as of 2010 (Che & Lewis, 2010). In the case of the X Window server, there is no "physical network", such as the internet, which keep the actors together at all times. It is more a case where the actors agrees on points of stabilization, this being a certain version of the software that is released, which I will explain in the empirical chapter. It is still a case where a stabilized set of practices are being maintained by a network of actors and intermediaries and is to some extent kept irreversible. Yet, within this network of actors two groups attempt to secede from the old habits and create new ways of getting things done, and in this process some of the actors have no choice but to follow either of the two directions depending on how interconnected and dependent these actors are to the ones leading way, these are "acted" by the network that holds them in place (Callon, 1991).

When studying standardization efforts of an electronic patient record system in Norwegian hospitals Hanseth, Jacicci, Grusot & Aanestad looked at the standardization process through the concept of reflexive modernization (2006). Reflexive modernization is a theory which describes a second modernization which happens after, say, a society has been modernized (Beck, Bonss, & Christoh, 2003). The theory focuses on the side effects or unintended consequences that come together with modernization processes which do not originate within the

social structures themselves, but which come from outside of the social structure. The side effects trigger the "affected" actors to act directly against the situation which they have been but in with the possible result of changing the initial structures. "While crises, transformation and radical social change have always been part of modernity, the transition to a reflexive second not only changes social structures but revolutionizes the very coordinates, categories and conceptions of change itself" (Beck et al., 2003, p. 2). The side effects are called reflexive since they are transmitted through the multiple networks and are finally reflected back onto what initially triggered them, and can be a result of the opposite of what was initially intended (Hanseth et al., 2006). A key feature in modernization is standardization, and one of the goals is to create order in a complex situation (Hanseth et al., 2006). What Hanseth et al. finds in their case, is that the process of standardization also generates such reflexive processes which undermine this goal, resulting in a disordering instead of ordering of the networks. An example from their case study when they looked at the implementation failure of a system called DocuLive, a project that aimed to replace the paper based patient record system, show how when trying to standardize by over emphasizing on criteria such as universality, uniformity and centralization in order to achieve alignment, stabilization and closure, the end result can turn out as the opposite of what they were trying to achieve.

> "When actors tried to stabilize the standard by enrolling more actors it became less stable. Attempts to improve fragmented records by means of one integrated electronic paper record made the records more fragmented. The complexity of DocuLive turned out to be on where the ordering efforts created disorder. The side effects triggered new ones, which again were reflected back on the origin. The standardization turned out to be reflexive and self-destructive" (Hanseth et al., 2006, p. 13).

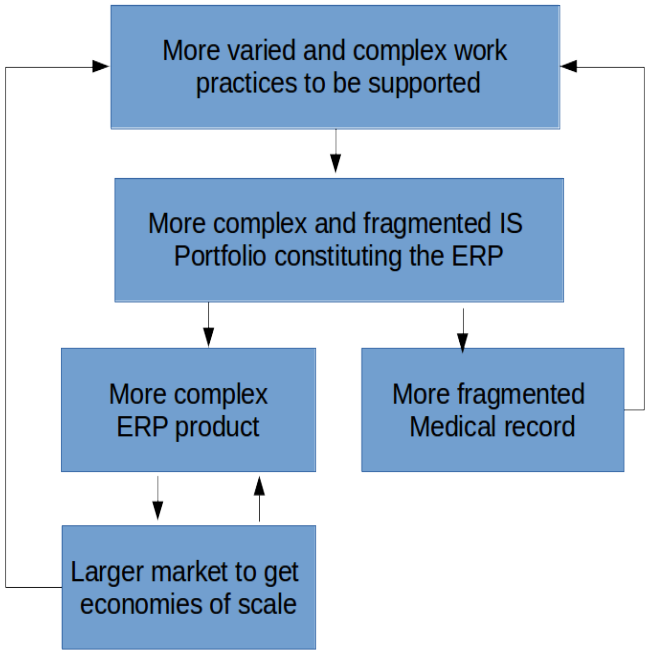


Illustration 4: The reflexive Standardization Process(Hanseth et al., 2006)

Illustration 4 shows how this process flows. I will use the concept of reflexive standardization to analyse my case, by keeping an eye open for processes that can be perceived as reflexive. If not confirming that there is any, it will at least serve the purpose of testing the concept empirically.

3. Methodology

When studying infrastructures, scholars from Science and Technology Studies (STS) propose several methodologies: Paul N. Edwards argue that one should consider how the different scales of force, time and social organization produces different pictures of how the infrastructures is being developed. He uses the history of Internet (in its infancy called ARPANET) as an example of how the story is being told differently from a micro, meso or macro scale of social organizations involved in the making of the ARPANET. The understanding of only the micro scale would not be sufficient enough to present the whole picture of this story and explains the need of a meso scale perspective by stating that the creation of large infrastructures requires large institutions with long lifespans, enormous political, economic and social power (Edwards, 2002). This statement was maybe more relevant in a time when communication between individuals and flow of information was somehow more limited than it is today. In post ARPANET time however, we have seen several examples of large world-wide spanning infrastructures being created by a distributed network of actors without the need of any significant political or economic power. A recent but very good example of this can be seen with the creation of the Bitcoin network, an attempt to create a Peer-to-Peer[9] Electronic Cash System that first started out with a couple of persons exchanging ideas on a mailing list in 2008 (Nakamoto, 2008). The computational power which this network representing is now, only 6 years of existence outperforming 500 of our supercomputers combined by a factor of 8 (Neuhaus & Polze Andreas, 2014). The building blocks of the Internet after ARPANET can also be seen as such an example, an infrastructure which was not necessarily "created" by individuals or organizations with much power, but rather built by someone, somewhere for one

[9] Peer-to-Peer – computer communication where each computer is communicating directly with other computers, as opposed to communicating trough a centralised server.

specific purpose, then picked and reused by others. This way, the adoption degree of each part that constitute the Internet determines which type of techniques that is being standardized, a sort of natural selection process which includes the possibility of *serendipity* where new uses of a part are later being discovered (Kelty, 2008).

The proliferation of different technologies around the world does create a vast mix of technology, politics and actors in diverse configurations that do not follow scales or political mappings (Latour, 1993). Scales are the spatial dimensionality necessary for a particular kind of view, whether up-close or from a distance, microscopic or planetary. The use of scales makes us imagine a particular human behaviour through a particular type of scale, the scale is however claimed and contested in cultural and political projects and, therefore, conjured by the "scale makers" themselves (Tsing, 2000). And since infrastructures are so diverse, they can be analysed in many ways, in the case of a technological system one way to analyse them could be to understand them as networked machines (Larkin, 2013).

Since I see similarities between how these networks I just have described are being created and how Free Software projects at large are being created, the Mir/Wayland controversy (as also pointed out in the introduction chapter about the TCP/IP protocol) can be viewed as a small infrastructural change performed by a network of actors.

When I decided what methods I wanted to use I attempted to take into account that scales do not always fit the configurations of technological systems and that they can also be conjured by myself or the makers of the scales (Latour, 1993;Tsing, 2000). As a result I avoided focusing too much on the different scales of political, economic and social power which Edward (2003) emphasizes.

I rather tried to looked at The X Window system as one technological artefact within an infrastructure. The artefact is functioning as a component or artefact which interacts with other

artefacts. If one component is then removed from this infrastructure, the other components have to adapt (Hughes, 1987). In order to identify the actors that were maintaining these artefacts or components and to create an assemblage of actors, I would need to trace the connections that interact with the artefact that is being replaced. This approach, known as sociology of associations or Actor-Network Theory or just ANT, is commonly used within STS (Latour, 2005). I decided to not fully make use of this technique, but rather borrow some of the interpretation methods practised by this approach when informing my readers about the objects and concepts explained in the literature part. The purpose of this approach was to create a network of actors that can be used in thinking about the case as one object of study or as a gestalt. I did conduct an ANT analysis per se.

Since ANT treats human and non-human actors equally in a heterogeneous network of actors, the non-human actors would in my case be the X window system, the Mir project and the Wayland project (Hanseth, 1998). The process of tracing associations was done through the interviews and through other sources (The interviewing methods will be explained in the last part when I present the material). After the actors were identified I examined their inscriptions. In the book *Laboratory Life*, we can read about how inscription devices are used to transform material substances to text through describing the substance on inscription devices (Latour & Woolgar, 1986). In my case, inscriptions were mailing lists, blog posts, meetings from Internet Relay Chat (IRC) logs and code comments submitted by the actors to the different code repositories[10] as well as the material from the interviews that I conducted. Since the non-human actors are written and maintained by the human actors, looking at code contributions helped me in following the author to the organization that the author represented.

[10] Code repositories for both the projects are free and open for everyone to observe and comment on. Wayland's repository can be found here, and for the Mir project here.

What I realized in retrospect by following this approach, however, was that when drawing too much on the concepts when analysing the material, I did actually create scales which were conjured by the concepts that I used (cautioned by Tsing). A compensation for this potential bias was therefore needed and addressed in the discussions.

At the same time, this is a study of a controversy which has not been resolved yet. Such controversies have been subject to investigation by STS scholars such as Pinch and Bijker known for their work on SCOT (Social Construction of Technologies) through looking at how actors give technological artefacts different meanings (Interpretive flexibility), and by reconstructing the problems each actor or in SCOT terms (relevant social groups) tries to solve with the certain artefact, they tell a story about how the artefact was formed (T. J. Pinch & Bijker, 1984). ANT, on the other hand, utilizes controversies in a way that lets the actors order themselves, rather than making the actors fit in certain predefined ordered system (Latour, 2005). The tracing of associations between the actors within the controversy which I mentioned above is then a way of recreating this order.

I made use of the latter method in the sense that I discovered my actors through inspecting the non-human actors, but as far as the ordering were concerned I used the recursive public concept. I did, however discover new actors through the interviews by asking my interview object about what persons they thought would be relevant to talk to; this led me to discover two new actors which I was not aware of prior to the interviews.

3.1 Limitations and ethical concerns

As a result of limited access, I had to interview my actors through different means of media, this being video, chat and email. When going through the material which these different approaches produced, I recognised a variation in how much content each method had produced. The result was a much more comprehensive collection of data from the Mir side of the con-

troversy which could affect the balance of my analysis. This also meant that I had to rely more on secondary sources when establishing material from the Wayland side of the controversy which could potentially damage my understanding of the full context.

Throughout my thesis I have also built most of my empirical data on electronic sources such as blog posts and forum comments. This raises some ethical concerns about how to refer to my sources. All of my sources in the empirical part is publicly available material and written by persons using their real name and not through pseudonyms. The actors which I Interviewed were all asked in beforehand about if they wanted to be cited by their real name or not, but all of them agreed that I could use their full name in the citations.

4. Empirical case and background

I was first introduced to the controversy surrounding these new display servers through the various forum and blog posts that circulated through those shores of Internet that I spent my time on in the period of 2012-2013. A big part of the empirical collection that was needed to be able to understand the process would therefore require me to go back in time and reacquire what I had been reading in the past, and from there, try to establish a chronological gathering of events, discussions and statements from the people and organizations that were involved. Since much of the discourse was being played out on social-media platforms, mailing lists and similar channels with a very rapid flow of information, much of the data was buried under a vast amount of new data, the task of digging through this was surprisingly harder than what I first imagined. Luckily most of the significant events and statements had been captured and retold by technology journalists, and reading through this coverage helped me a lot when mapping the events to a time-line. I was also during this process re-introduced to more actors, some of which I later decided to interview. I will in this chapter present the history and events

that led up to the controversy in a chronological order and explain the case in more depth by drawing on some of the empirical material. I will also present you to the actors which I interviewed.

4.1 Brief history and chronological collection of events

The first seeds of the X window system can be found in the V operating system or V-System developed at Stanford University in the period of 1981 to 1986 (Theimer, Lantz, & Cheriton, 1985). The window system used in this operating system (then called the W Window system) was in June 1984 used as a reference when designing the initial release of the X Window System (X1) by a project at Massachusetts Institute of Technology called Project Athena (MIT, 2014). The X Display Server did despite arguments against its efficiency (Gajewska, Manasse, & McCormack, 1990) become widely used among UNIX like operating systems (Linux, BSD, Solaris and OS X versions prior to OS X Leopard(10.5) used by Apple computers.) and eventually became a *de facto* standard. In 2014, it was still the most used display server protocol for these types of operating systems. After about 25 years of use the X window system had grown to a comprehensive project, and its complexity seemed to cause problems for projects that wanted to build on top of it (Shuttleworth, 2010). When I asked my interview objects about their experience when working with the X Window System, Martin and Oliver (which I will introduce to you below) answered: “X is just too old for a modern setup, too many legacy issues” (Gräßlin, 2014) and “From an architecture perspective, it was sort of like putting a fifth band-aid on top of a stack of band-aids” (Ries, 2014). This experience seems to be reflected throughout the community; Illustration 5 show a meme that has been circulating around in the Free Software community where xorg.conf refers to a file which is being modified when configuring the X Window System and reflects in a humorously way how the com-

munity feel about X. The situation explains not only a lack of flexibility the actors have when working with X, it also explains how the growing complexity of the project have created a black box, which the actors are unable to open, resulting in the need for a new box.

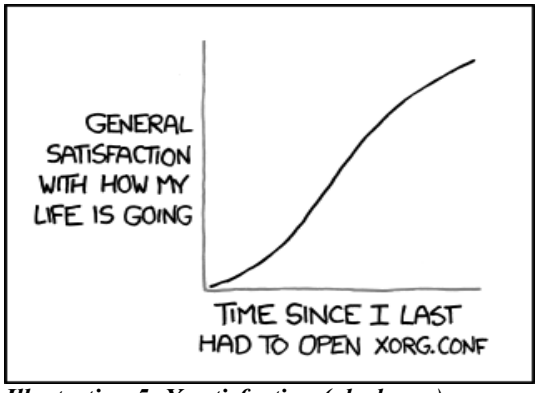


Illustration 5: X satisfaction (xkcd.com)

In 2008, Kristian Høgsberg (at this point working for the company Red Hat) had started working on the new display server protocol called Wayland with the purpose of solving this problem (Høgsberg, 2008). In February 2012 Høgsberg released the first version of the Wayland protocol definition along with an implementation called Weston (Høgsberg, 2012). Wayland gained attraction among projects that at this point were supporting[11] or using the X display server protocol, some of these included:

- Enlightenment (Desktop Environment)
- MESA (collection of graphics libraries used in 3D computer graphics)
- Sailfish OS (Operating system for smart-phones)
- Tizen (Operating system for smart-phones and tablets)
- GNOME (Desktop Environment)
- Qt (application framework for developing graphical user interfaces)
- GTK+ (application framework for developing graphical user interfaces)
- KDE (Desktop Environment)
- GENIVI (In-vehicle infotainment system used in cars)
- Ubuntu (Linux based operating system)

[11] To support another software project either means to implement code to make the two different software programs talk together, or making plans for such changes in future releases. The support can also include labour or financial means of support

In March 2013, however, the company Canonical behind the operating system Ubuntu, who previously had expressed support for Wayland (Shuttleworth, 2010) announced that they were developing their own alternative to the X Window system, a display server called Mir. In an email to the Ubuntu developers Director of Engineering in Canonical Oliver Ries wrote:

> "After thorough research, looking at existing options and weighing in costs & benefits we have decided to roll our own Display Server, Mir. None of the existing solutions would allow us to implement our vision without taking major compromises which would come at the cost of user experience and quality. We will be running sessions at the UDS[12] to discuss questions and take feedback" (Ries, 2013)

This announcement led to a debate around to what extent it was necessary for the Free Software community to support two different display servers and the company was criticised for adding unnecessary fragmentation to the Free Software ecosystem (Brown, 2013). Software fragmentation is something that occurs when you have several actors creating their own solution to a common problem, this often results in less interoperability and the need for supporting more than only one project.

In responding to this critique, Canonical employee Robert Ancell who was working on the Ubuntu project wrote in a blog post about how this additional display server actually does not matter that much. He argues that there is another layer in the technology stack that is more important to developers and users, namely the application framework[13] used in making applications. He wrote: "display server doesn't matter much to applications because we have pretty good tool-kits that already hide all this information from us" (Ancell, 2014). Yet, it

[12] Ubuntu Developer Summit

[13] Application frameworks is a set of libraries of code used when making an application with a graphical user interface. Qt and GTK+ are some examples of such. They are also often refereed to as just tool-kits.

seemed to matter for some people. In a response to this blog post, Martin Gräßlin who works on a project called KWin[14] wrote about how this situation causes problems for his project:

> "Canonical created a huge problem by introducing another Display Server and it's affecting all of us and they are still in denial state. It's not a simple the toolkit will solve it. It can cause issues everywhere and that affects the development and maintenance costs of all applications. My recommendation to application developers is to never accept patches for this mess Canonical created. If they want that fixed, they should do it in their downstream patches. Distro specific problems need to be fixed in the distros. I certainly will not accept patches for the frameworks and applications I maintain. This is not for political reasons as it's so often claimed, but for technical, because I cannot test the patches" (Graesslin, 2014).

Also Aaron Saigo from the KDE project wrote a long post with arguments as to why it matters for different reasons: "Having different display systems will negatively impact different groups of people, and at the end of each of those chains are users who are affected and therefore care. If we care about free software, then we need to care about those who use free software*"* (Seigo, 2014). To briefly sum this up, we have the X Window system which all the actors agree is or will be obsolete in the future – I have not managed to find any that disagree on this. The creation of an alternative to X which is Wayland, led almost all of the actors to work on supporting this new alternative. But one of the actors decides to create their own alternative to this joint effort, Mir. This decision seemed to have lit a fire within the Free Software community, causing debate between representatives from Canonical and representatives from other parts of the community working on other projects.

The debate raises the following issues and questions which I thought would be interesting to investigate when analysing the material further: In which ways are the actors involved in

[14] KWIn – a window manager for the desktop environment KDE

this controversy connected? Are the actors connected only trough the technologies that they are using or are there any other associations? The second research questions, looks at what type of considerations the actors need to take into account when making decisions that can lead to a controversy like this. Are there any political motivations behind them or any other types of power mechanisms that is lurking underneath the surface? I also decided to focus on how their decisions could affect other projects and how the actors responds to such. The overall theme however, is about how the process may create a new standard, or whether it is a failed standardization attempt. I will in the last part therefore focus more on the case in a broader context with focus on how the case relates to standardization processes.

4.2 Contacting the actors

In order to acquire additional data for my analysis I decided to conduct interviews with some of the actors involved in the debate. I wanted to talk with both sides of the controversy in order to observe from both viewpoints (Venturini, 2010). Since the majority of the actors seemingly was supporting the Wayland project it became crucial for me to get an interview with someone from Canonical which is the company behind Ubuntu and the Mir project. Since the production of Free Software projects are in many cases globally distributed, I had to propose other means of communication than meeting in person. I initiated contact with Canonical through an email and after some exchanges I was referred to Oliver Ries who worked as the Head of Engineering, the department responsible for the development of Mir. Oliver was also the person that sent out the initial announcement about Mir in March 2013 mentioned above. Oliver was positively interested in participating in an interview and we decided to schedule a meeting on the video chat service Google Hangouts as he was situated in Utah, USA.

Amongst the Wayland supporters I decided to contact both the GNOME Foundation and the KDE Community. Both of these actors are developing their own Desktop Environments

and have both indicated support for the Wayland project through public statements and code implementations (GNOME, 2014). I ended up getting in contact with Martin Graeßlin from the KDE project who was positive towards answering questions. Martin had also been active in the Mir/Wayland discourse on both his blog and the social media platform Google plus. At the time he worked for a company called Blue Systems in Bielefeld, Germany who delivers KDE software, and was in his position dedicated to work on KDE related software such as Kwin, a component in the KDE project. We decided to exchange questions and answers through email.

Additionally I contacted the founder of the Linux Mint project and asked him if he was interested in a talk, he agreed on a meeting on the Internet chat platform IRC[15]. The Linux Mint project is known for delivering a mixture of different combinations of Desktop environments and had at this point not expressed any opinions in the debate. I thought that the Linux Mint project would add a valuable perspective since the project seemed to be agnostic in terms of adopting either of the two display servers into their project. This perspective would also provide a greater contribution to the second research question (II) which do not necessarily limit itself to the controversy.

[15] IRC – or Internet Relay Chat is a communication platform, often used by software developers to discuss their work.

5. Empircal analysis

This chapter will provide analysis and discussions of the issues and questions that were raised in the previous chapter. I will begin with looking at the potential underlying causes of the controversy and investigate the question about how the organizations are connected through the technologies that they share and cooperate on; the first section will serve as a foundation for understanding the process defined by Prof. Steven Weber as the following:

> "The open source software process is not a chaotic free-for-all where everyone has equal power and influence. And it is certainly not an idyllic community of like-minded friends where consensus reigns and agreements is easy. In fact, conflict is not unusual in this community; it's endemic and in a real sense inherent to the open source process. The management of conflict is politics and indeed there is a political organization at work here, with the standard accoutrements of power, interests rules, behavioural norms, decision-making procedures, and sanctioning mechanisms. But it is not a political organization that looks familiar to the logic of industrial era political economy" (Weber, 2004, p. 3).

5.1 Why a controversy?

Weber (2004) recognises a different logic within the political organization which Free Software organizations represents, a logic which does not look similar to what conventional knowledge can explain. So, in order to reveal some of this logic, I will fist look at how the organizations are connected, which is my first research question.

In creating a picture of this political organization which Weber calls it, I will make use of the concept presented in the literature part of a recursive public. According to Christopher Kelty "A recursive public is a public that is constituted by a shared concern for maintaining the means of association through winch they come together as a public." (2008, p. 28). In this case, the actors that uses the X Window system represents the recursive public, the artefact represents one of the means, which people from different organizations maintain and care for. Another way to think about this is to abstractly view the artefact (software project) as a written down idea which describes how a system should work in order to achieve its goals, which are shared by many and improved upon by some.

To estimate how many people or organizations this particular common represents is not a simple task, one measure, however, can be to look at the number of contributions made in the code repository for the project. The implementation of X has during the last 10 years received contributions in the form of code changes from 1,117 different people (Ohloh.net, 2014). These numbers give us some idea of how many people or organisations this particular "means of association" hold together in maintaining this recursive public.

Prior to the announcement of Mir and Wayland, all my actors were connected together through this common "mean" since they were all in some way or another leveraging the X Window system project, they all had an interest in supporting the project and were all benefiting when its code improved. All the actors were therefore dependent on the X Window system in order to exist in their current form.

But they were also connected through other means of associations. Each layer of technology these projects utilizes can function as means of associations to other projects. Free Software projects may share hundreds of software libraries as explained in the literature chapter with the OpenSSL and Firefox web browser example. I have in illustration 6 created a figure

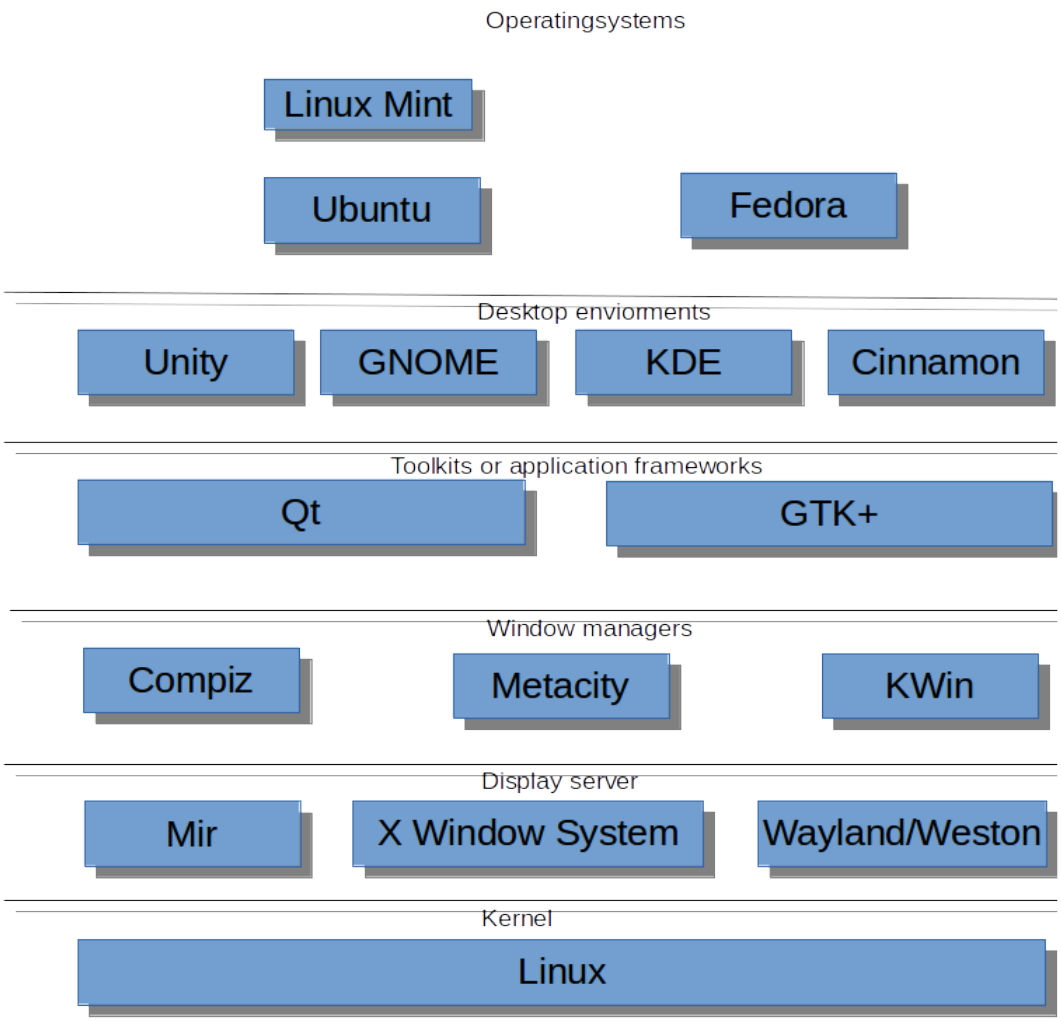


Illustration 6: Layers of technology

which abstractly show some of the technical layers which are being used by a typical Linux operating system. Each layer contains one or more projects which all do the same thing or solves the somewhat same problem but in different ways. If I were to create a new operating system myself I could mix my own set of dependent parts to build it upon. In the case of my actors, they are therefore still connected through leveraging of the Linux kernel project at the bottom of the figure, and most likely a lot of more parts if we were to create a more comprehensive illustration involving all dependencies these projects are constituted on.

When developers, my actors included, refer to their dependencies or projects that are dependent on their own project, they uses the terms upstream and downstream. This term can be used both as e verb and a noun and refers to which way the changes flows between the projects. Illustration 7 show this type of relationship between X Window System, Ubuntu and Linux Mint. For Ubuntu the X Window System is one of their upstreams, similar for Linux Mint Ubuntu will be one of theirs. The changes that go into the code of Ubuntu will therefore affect Linux Mint, thus is Linux Mint talked about as one of Ubuntu's downstream projects (Fedorawiki, 2014).

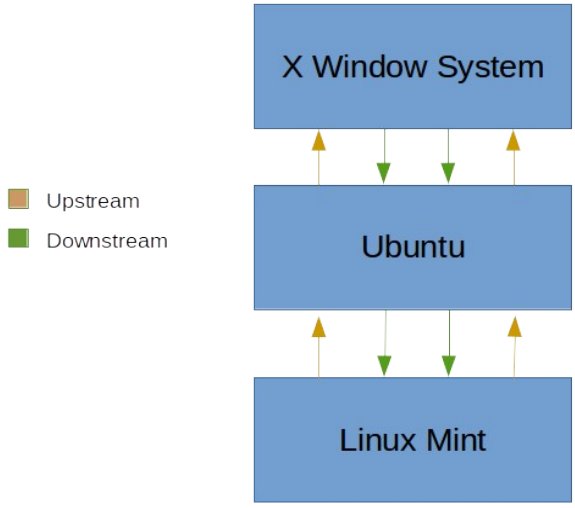


Illustration 7: Relationship between upstream/downstream

A result of this relationship, however, is that if a set of changes goes into one of these projects, the developers will be able to draw out implications for other projects in this layering of parts. This explains why there can be disagreements and controversies among the actors. But as Kelty points out, the layers are not necessarily limited to technical layers, it might as well also be law, regulations or physical constants (2008).

So, in order to identify a non-technical layer in the Wayland/Mir debate I decided to look at how the different actors referred or talked about their users, which also seemed to be a central topic of the debate.

Even though the different actors are creating products that are being used by different groups of people, they seem to refer to their users as one group, not “our users” as you would

expect from traditional marketing communication in companies. Here from Roberts note on why the display server don't matter: “I hope I've given some insight into the complex world of display stacks and shown we have plenty of room for innovation in the middle without causing major problems to the bits that matter to users.” (Ancell, 2014).

Also from Aarons response:

> “Do users care if their display system rocks? Yes, they do. It is a critical piece in terms of reliability (e.g. stability) and performance when using a device. It defines what can work well, what will work poorly and what can't even happen at all when it comes to windowing systems, application performance and security. The display system places an upper bound on how great the desktop shell and applications that run on top of it can be. We've worked with the limitations of X11 for a long time, and we owe every user to deliver an awesome display server, and that is less likely to occur with multiple projects dividing efforts” (Seigo, 2014*).*

In both of these statements, it seems like they are talking about users as if they share the same group of users.

If some actors view their users as someone that are not only using their own products, but as a big group of users who is shared by all the other, it would mean that if the total user base is another layer for the recursive public to draw out implications from, actors that are progressive in terms of changes should expect criticism if their changes can affect parts of the total user base in some way or another. When I talked to Oliver from Canonical about the issues of fragmentation he told me: “I'm always struggling with the claim that we are fragmenting it because everybody now has to deal with Mir. If you think it sucks, then don't use it right now.”. (Ries, 2014). By suggesting not to use Mir as a solution to the problems raised by Seigo about Mir, he might then be implying that the change should not affect others than the users of Mir. If this is the case, a different perception of this non-technical layer (who are the

users) could possible be an underlying cause of the controversy. This also highlights why it matters for the Wayland supporters, but not so much for the Mir team which was another central topic of debate.

I will in the next section go into more detail about how these implications are being considered and how they also change over time due to the development cycles that these projects has to work with.

5.2 Technological considerations

I have described how decisions made by one actor can trigger other actors to look at implications for the Free Software community as one common group. I also touched upon how projects choose between different layers of existing technologies when creating or putting together their product. In this section I will look more into the latter and try to reveal some of the mechanisms that is at work when Free Software organizations manage their projects. I will discuss what type of considerations the actors had to make when they made their decisions, and lastly, to what extent their considerations were determined by decisions made in the past, or put in another way, to what extent they are determined by the dependencies that the project is built upon.

When I interviewed my actors I tried to talk about how they where managing their technologies as their circumstances change over time, and how they decide which technologies to implement in their projects.

When I asked Oliver from Canonical how they choose and manage upstream[16] projects he explained:

[16] Upstream – is used in software development to refer to the original author or maintainer of a software project. As an example: Linux Mint is based on much of the codebase that Ubuntu is made of. The Linux Mint would therefore refer to Ubuntu as one of their upstreams.

> “So that is the big challenge for a Linux distribution such as Ubuntu or Red Hat or similar right, the whole variety of open source projects and those are what we consider the upstreams, they create something, then you as a Linux Distribution take that upstream and refine it for whatever that means and make a distribution out of that. And that is sort of challenging and different from traditional software engineering because you do not really have a lot of control over those upstreams” (Ries, 2014).

In tackling these challenges concerning lack of control he said: *“*dealing with it typical means either, lets see if we can find a good example: You pick the latest version of an upstream and then you look at the latest official release and the latest development release, and then you cherry pick the fixes or the new features, so this is typically how a Linux distribution deals with upstreams.*”* (Ries, 2014). This was also how Clement Lefebvre from the Linux Mint described how the process usually went on for them, which was not very surprising considering the way the projects are built on top of each other as shown earlier. If there are no big changes in the upstream code, the need for any significant considerations is less stressed. The organizations then just look at what are being changed in each release, and then decide to implement the changes in their own release or not.

When looking for an example of how the process of deciding can play out I managed to find an IRC log from a meeting held by the Engineering Steering Committee of the operating system Fedora. Fedora is an operating system that uses the desktop environment GNOME which are implementing support for Wayland. Fedora would here either decide to follow the path that GNOME has chosen, which is to implement Wayland support, or do something else. Here is a log from the IRC meeting which were held 7. May of 2014 together with an interpretation below.

```
17:50:20 <mitr> #topic #1306 F21 System Wide Change: Wayland - https://fe-
doraproject.org/wiki/Changes/Wayland
```

```
17:50:23 <mitr> .fesco 1306
17:50:24 <zodbot> mitr: #1306 (F21 System Wide Change: Wayland -
https://fedoraproject.org/wiki/Changes/Wayland) — FESCo - https://fe-
dorahosted.org/fesco/ticket/1306
17:50:59 <pjones> so very +1
17:51:03 <dgilmore> +!
17:51:05 <dgilmore> +1
17:51:08 <mitr> +1
17:51:16 <notting> +1
17:51:27 <mattdm> +1
17:52:07 <t8m> +1
17:52:50 <mitr> #agreed Wayland change approved (+6)
```

Each line can be divided in three parts where you first have the time of when each message is posted. Second you have the username within the two brackets, and third you have the message. Each user name can be translated to Bill Nottingham (notting), Dennis Gilmore (dgilmore), Peter Jones (pjones), Matthew Miller (mattdm), Tomáš Mráz (t8m) and Miloslav Trmač (mitr) which all are members of a rotating Steering committee, a group whose mission is to handle the process of accepting new features from their upstream projects (Fedoraproject, 2014). The user <zodbot> is a programmed non-human user who is recording the meeting and keeps track of the votes. In the example above zodbot first posts a link with information of the suggested change, which in this case is to implement Wayland, each member of the chair then votes with either +1, 0 or -1. In this case all members voted for in a process lasting for 2 minutes. The matter has most likely been discussed in beforehand on other channels and the purpose of the vote can be viewed as a formal way of dealing with changes, but it indicates that projects that are dependent on certain technologies easily follow their upstreams path, which in Fedoras case is GNOME, with no considerable controversy(considering the votes 6 to 0). Path dependence is in Fedora and GNOME's case written down in the code they use which make the decision easy given that the changes do not cause any major problems to their project.

But if we go back and look at Linux Mint, which is built on top of Ubuntu, I found another scenario. Since the development of Ubuntu is being done continuously, the stability of the software varies throughout the year. If a lot of new features are implemented into Ubuntu there will be a lot of new features that have not been fully tested and the stability will go down, it will however stabilize over time as the users report back to the developers about what is not working, which the developers then can implement. This "permanently beta" situation is not a phenomenon distinctive to Free Software projects, similar development cycles exists within web development and other non-free software projects (Neff & Stark, 2002). But the need for collaboration and timing seems to me to be somewhat unique or more important in Free Software projects. Many Free Software projects solve this problem by planning out dates for stable versions of their software project. Ubuntu as one example, releases two stable versions every year, usually in April and October. And every second year they release a version with longer support which aims to be more stable and less experimental than the twice a year releases. In figure 8 you can see how this cycle is planned out (UbuntuWiki, 2014).

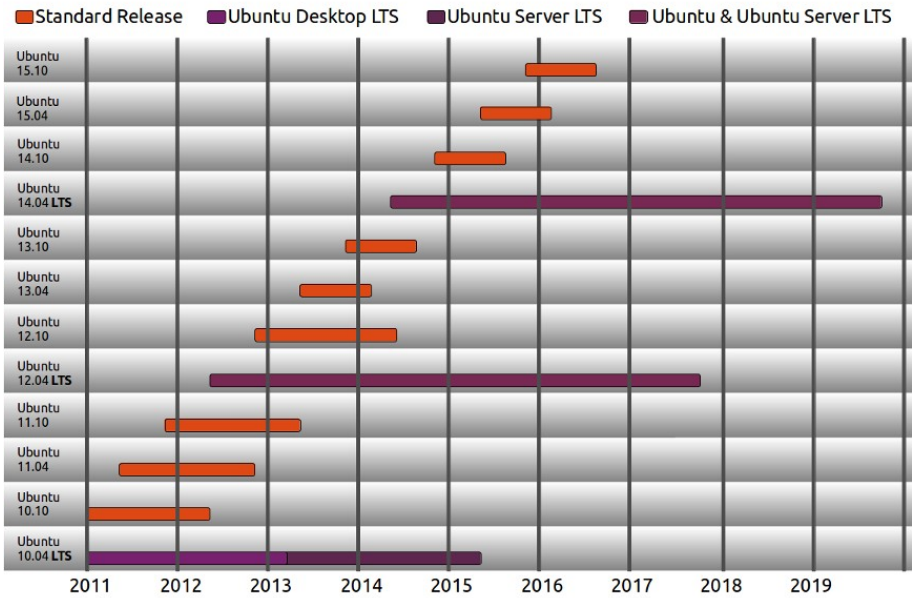


Illustration 8: Ubuntu release cycle (UbuntuWiki, 2014)

Linux Mint, as mentioned before, build their product based on what Ubuntu have already created and therefore schedule their stable releases a few months after the stabilization of Ubuntu. Any major change to the Ubuntu code would therefore affect the Linux Mint project when they are preparing for a new stable release.

When I asked Clement about how Ubuntu's Mir decision was affecting their project he told me: “We're considering sticking to the Long Term Support for the next 4 releases. So that would postpone any need for us to look into this for a whole 2 years.” (Lefebvre, 2014). So this would be an example of how the usual process explained earlier by Oliver and Clement is interrupted and requires the project to make further considerations. In this case the Linux Mint developers decided to skip three releases of Ubuntu and freeze it for two years which usually is done every six month when a new release of Ubuntu is released. The Linux Mint developers thus postponed their need to act on the changes for two years. In relation to path dependence this could be viewed as a way of taking over the control of the direction that they are headed towards. But only as a temporary solution considering that the software needs to eventually change in order to stay alive (Hanseth et al., 1996). Staying alive in the context of Free Software domain usually means to stay relevant and used by the actors who keep the code alive by use, modifying and redistributing of the code. But it does show how big impact changes in upstream code can have for dependent Free Software projects.

For Ubuntu when looking at Wayland as a potential upstream, considerations led to another way of dealing with this:

> “So when we where looking at Wayland in it self as a protocol and trying to see if it matches our designs and like, you know we have or Mark has his vision in his mind of what he wants to do for the phone and where computing or personal computing should be in two years and four years and so we have certain designs that are partially public and partially are just in Mark's head still

> as a part of his vision. So, we took that sort of as a set of requirements, and we are looking at what is already implemented and what is sort of defined in Wayland and how easy it is for us to fulfil our requirements from a design perspective. So that was one of the first criteria that we applied, our finding was that while a lot of the things were there, we thought that a couple of things might be missing, and then it is an open-source project, so what we could do is, you could collaborate with the upstream development team and get it in, but a problem that of few that we had or still have with Wayland is that it runs into similar issues like X" *(*Ries, 2014*)*.

So in this case the engineering group receives a set of design requirements from the decision maker (Mark) which is sponsoring the development. The engineering group then looks at which projects are available and finds Wayland, which have some limitations that are similar to the limitations that they have experienced when working with the X Windows System. The limitations are further explained as:

> "if you think of where X is running right with a use case fulfilment that is large, that is mostly because X is very extensible right, lots of extensions, anybody can write to X and then you can put it onto your microwave or whatever. But, the extensibility also comes with a price which is a lot of incompatibility" (Ries, 2014).

And "So, with Wayland we feared a similar situation, or our concern was that something similar would happen because Wayland stately I think, says it wants to be a multi-purpose Display Server." (Ries, 2014). When looking into the future, based on what they know about Wayland today, they fear that Wayland would run into the same problems (which I will discuss later) that the X Window system ran into. In order to avoid this scenario they decided to create their own path instead of following the path that their dependencies suggested them to follow, which were to use Wayland.

To summarise this, my findings show that Free Software projects have at least three different ways of managing the unstable environment of being reliant on other projects, or to manage the path which they are determined towards as a result of building their product on top of other projects:

1. Create an alternative project if the path is contradicting with the needs of the software or goals of the organization in charge, as shown when Ubuntu decided to create Mir.
2. Decide to not implement changes in order to postpone the need for making a decision, as Clement from the Linux Mint project described.
3. Implement the changes, and adapt to the changes, or as Oliver expressed: include the changes into your own project and refine them, which is how they typically manage upstream projects.

So, the concept of path dependence seems to play a role in terms of their need to take action when the path is changed, the organizations still have the possibility to steer away from the suggested or decided path. In Linux Mint's case at the expense of an outdated code base, and for Ubuntu at the expense of what it takes to create an alternative. This indicates that the notion of path dependence does not play a significant role from a code perspective even though the projects are built on top of each other and thus dependent on each others work. It played a much lesser role than what I first anticipated before conducting the interviews. That said, other factors such as economic, labour and political might play a bigger role in this than the code itself, which will I will discuss in the next section.

5.3 Politics and power of code

Although the development of Mir had already started at the time this interview took place, Oliver and the Ubuntu team did not seem to exclude the possibility of changing their minds and use Wayland instead. If the implementation of Wayland would turn out to work for them

then he saw no reason for them to not use it: “At the end of the day, code wins, right. If the implementation is just super awesome and if the architecture is great we probably would be OK to take that overhead and friction in collaboration because there is a lot of work already out there that we could base on” (Ries, 2014). This brings out not only how strong mandate the code itself seems to have to when actors consider technologies, but it also highlights how some developers have a pragmatic attitude to the technologies they are working with. If one technology is better suited to fulfil a purpose that their own project also tries to fulfil, there seems to be no reason to not use the better one, even if it means giving up the development of their own project. This might sound like a bold observation in favour of technological determinism, but I want to emphasise that a certain technology is only viewed as better relative to how many problems the technology solves for the particular actor. In SCOT as an example, each relevant social group have various types of problems they want to solve, the solution to their problems are being reflected in the design of the artefacts that are being created. Thus, the artefacts will also be shaped by the sociocultural norms and values that the actors hold (Pinch & Bijker, 1984). Another concept which Pinch & Bijker focuses on is the concept of *design flexibility* (or *interpretative flexibility*)*;* how the artefacts has different meanings and interpretation depending on which actor you ask, such an observation can be found when looking at how my actors value the various projects differently. When Oliver says that something works better for Ubuntu, a project that aims to be a modern operating system which can be used on computers, smart-phones and tablets, this “better” does not seem to be the same for Clement from Linux Mint, a project which aims for a stable and more conservative operating system focusing mostly on computer desktop support. “When Mint goes more conservative, it will work better, but will be less exciting to people who want to run the very latest” (Lefebvre, 2014). These different set of values are also reflected by the total prolifera-

tion of Free Software projects where as an example the GNOME desktop environment focuses more on simplicity while KDE on functionality. “The philosophies are distinct, with GNOME tending towards minimalism and KDE towards what might be called 'completism'” (Byfield, 2014).

So, the considerations that the organisations need to take seems to be focused on choosing the right technology that would best suit their project in delivering what the projects aims for, or in SCOT terms “Solving the problems for the relevant social groups” (Pinch & Bijker, 1984). If the technology is not available, they create it themselves if they have the required resources.

This pragmatic way of solving problems for their own projects would create a more diverse collection of Free Software projects and does not seem very compatible with making standards that can be shared among the actors. When I asked my interview objects about standardizations and diversity Martin told me: “I consider the needless diversity as a huge problem for Free Software in general” (Gräßlin, 2014). Clement had a more two sided view,

> “Well, I like when things work. So I suppose I like both. Without standards you couldn’t run Cinnamon outside of Linux Mint or KDE in Ubuntu etc. too many things would break. If everybody coded their thing on their side and didn't care about standards, in a matter of years we'd lose compatibility. With that said, I believe in getting things done.” (Lefebvre, 2014).

Here, Clement points out the importance of standards in relation to his project and how standards enables a variety of *use cases* for the project that he is working on. At the same time he puts some emphasis on the power of “getting things done” which I interpret to mean being pragmatic in terms of creating something without focusing too much on cooperation with other actors (e.g. agree on a standard way of doing things.) Oliver's response was also twofold

> "I think standards are important, at the lowest applicable level. Standards and diversity sort of conflict in varied tensions. Where you have things standardized you wont have much diversity. I think as an open source community it is important for us to standardize on low level components like kernel interfaces or even like, you can go higher and talk about qt which is a standardized set of API's that we are building on.(..) So I think this is my stands on standardizations and diversity. Standardization is important on a lower level and if you do it this way it allows you a lot of diversity on top of that." (Ries, 2014).

Oliver also recognizes the tension between the two, but at the same time he suggest to agree on standardizing on only some layers on the technology stack. The kernel layer is one of the layers he suggests being standardized, which is the lowest layer in this case. Also *qt* which is found in the tool-kit layer I illustrated back in illustration 6. So by creating Mir, the Ubuntu team is expressing this view by using their voice through the technology they create instead of arguing in beforehand about how they think the infrastructure should look. This type of activity is also observed by Kelty and was one of the characteristics he used when describing the recursive public. "geeks use technology as a kind of argument, for a specific kind of order: they argue about technology, but they also argue through it. They express ideas, but they also express infrastructures through which ideas can be expressed (and circulated) in new ways" (Kelty, 2008, p. 29). The idea or argument which the Ubuntu team expresses can be interpreted in two ways: One way is to look at it as an argument against the need for a multi-purpose display server standard, which Wayland represents. Another way can be to look at it as an objection to the process itself; the overhead and friction in collaborating which Oliver described caused too much insecurity for the Ubuntu team.

From a *de facto* standardization perspective this case also brings out some of the dynamics in the creation of standards which is not ruled by a central body. The process described also

share some similarities with how the Internet Engineering Task Force (IETF) worked out standards through their Requests for Comments (RFC) system which Steve Crocker developed in 1969 to help specifying the standards of ARPANET (Hauben, 2010). IETF is however an organization that works out formal standards, while Free Software standards are more often formed through the cultivation, distribution and use, *de facto* standards are thus being formed in a more organic way than say standards worked out by a centralized group of people. Principles such as putting the code above other means of power seems however to be shared between the two different processes, as expressed at an IETF conference: “We reject kings, presidents and voting. We believe in rough consensus and running code” (Clark, 1992). So, in such an environment, code can also be power, and in this case the power is used in arguing how the infrastructure should look.

This also means that for Canonical, a strong commercial actor with incentives for making strategical decisions which will help them to stay alive as a commercial organization, the motivation behind creating Mir could also be motivated by the benefits of controlling a part of the infrastructure which they build their product upon. Here from Oliver:

> “We have a goal and that's sort of selfish but that is because we are a company right, and like RedHat and everybody else who wants to make money we try to not incur overhead that would not get us to our goal, so we thought that collaboration was a bit difficult.” (Ries, 2014).

The notion that code can serve as a way of enacting power or control is not an unfamiliar view, especially not within the domain of proprietary software. Free Software is in contrast usually considered as a counter measure against various types of power. Prof. Lawrence Lessig expresses this view in his book *Code v2*. “*(..)* whatever side you are on in the 'free vs. proprietary software' debate in general, in at least the context I will identify here, you should

be able to agree with me first, that open code is a constraint on state power" (Lessig, 2006, p. 139). His argument is followed by examples of voting systems used in presidential elections, and how some parts of such systems should be open in order to allow transparency hence diminishing the risk of abuse by those controlling the system.

But, in the Free Software context, the potential power abuse which Lessig talks about is non-existent due to the licensing and open design of Free Software itself. Nonetheless, there still seems to be power mechanisms at play. In Bergquis & Ljungberg's study, they also looked at power mechanisms, and found that motivations behind getting contributions out in circulation were a way for Free Software actors to guaranteeing on the quality of the code, and drew a parallel between this activity and how contributions in research communities can be viewed as gifts in the form of scientific knowledge (2001). When Oliver and his team looked at the process of the Wayland development they looked at how easy it was for contributors to "get something into the Wayland project". They concluded that it would not be easy for them to achieve their goals by following the process that they observed. This evaluation show that there are a different weights of power present between the persons in charge of a software projects and its external contributors. As one of the responsibilities of being a maintainer of a project is to accept changes (patches), this mandate can also be used to control what types of changes goes into a project by rejecting the changes. The maintainer will therefore to some extent be able to influence the design process of the artefact which is being created. An actual example of this use of power can be seen in an accepted patch submitted by the Mir team to one of Intel's X driver projects. Intel is one of the supporters and funders of Wayland, and the project itself is developed through Free Software principles. The patch was first accepted by the developers, but then rejected with a note saying "We do not condone or support Canonical in the course of action they have chosen, and will not carry Xmir patches

upstream -The Management" (Wilson, 2013). The note signed by "The Management" indicates that someone higher up in Intel's organization chooses to act to prevent the patch from being accepted in order to control the development most likely due to political reasons. Oliver's interpretation of this event was: "That was sort of the funny thing right where you have the collaboration on the engineering level someone responsible for the code, thought the patch was OK, but then when it sort of reached the corporate political level, all of a sudden they came to burn it hot" (Ries, 2014). This example can also serve as an example of how the different scales of social organizations which Edwards (2002) focuses on, is important to be aware of when constructing an image of how artefacts or infrastructures are being created.

Both older and recent studies have, according to Bijker, avoided the focus on power in the process of technological development, not because the phenomenon is not identifiable, but since such explanations do not offer very much insight in terms of explaining the actual process (1997, p. 11). The observation of the use of power which the Intel representative demonstrated does at least for my case serve as something that is tangible and identifiable when trying to observe how the artefact (in this case Mir) is being created. In order to prove this we need to know what the differences in outcome would be if the patch was accepted or not, or rather what it would mean for the artefact. A simplified version of the purpose of the patch was to enable the possibility of running old applications created for the X Window System on top of Mir when using a video-driver[17] produced by Intel. It would in other words determine in what way two pieces of software interact with each other, which would have made a great difference in terms of how the artefact could have been designed.

My argument is therefore that the artefacts themselves end up as a reflection of how these organizations are connected, together with the values the organizations inhabit, and if there

[17] Video-driver: a code library that is designed to operate a hardware device, in this case the hardware that is connected to video monitors.

are power mechanisms present, they will also be reflected in the design of the artefacts. This also means that the artefacts can be used as tools to shape the Free Software infrastructure, both in terms of software architecture but also the relations between the actors. So, although some scholars have made observations about how the Free Software community is more or less political agnostic in the sense of denying any traditional political association (Kelty, 2008;Colman, 2004), there seems to be politics at play between the actors inside the community.

5.4 The result: A more complex technology layer

If we try to scale out, and look at the Wayland project as a collective effort by the supporting actors to move towards a new standard, the project has succeeded by consolidating almost all of the actors who uses the X Window system. But is it a success in terms of a standardization effort if the goal was to reduce the complexity represented by X? As Oliver explained, the main reason for them to create their own display server was because they thought that the Wayland project were heading in a direction which would cause much the same problems as the X Window system. The problems with X described by Oliver were “maintainability and then the extensibility were like really key considerations in making the decision to move away from X” (Ries, 2014). This view is shared by the Wayland team, on their website they point out what they think is the problem with X: “The problem with X is that... it's X. When you're an X server there's a tremendous amount of functionality that you must support to claim to speak the X protocol, yet nobody will ever use this.” (Wayland, 2014). The Ubuntu team did not feel comfortable that these problems were addressed by creating another multi-purpose display server. The problems they recognised were however not limited to the definition or implementation of Wayland itself, but to the process of collaboration. The process was

construed by Oliver to include unnecessary friction, which the Ubuntu project could not afford to deal with in order to solve their problem, he explained:

> "So, we looked at mailing lists and we looked at submissions and how the process was, how the community was engaging with contributors and how easy it was for an external contributor to get something into Wayland. And we didn't feel comfortable. (..) If we go into a project and say 'This is what we would like to do' our fear was that we would end up in endless discussions about how this feature that we would really want for Unity would apply to KDE and Gnome and all the other Desktop Environments and we would just spend a lot of time in, there would be a lot of friction in getting things done into the project, that friction would not necessarily benefit us" (Ries, 2014).

So, after evaluating the process in which way the Wayland project handled code submissions from the different actors, they concluded that the process would lead to discussions about how their changes would cause implications for other projects like KDE and GNOME. These discussions were considered as friction which would slow down the process of solving their initial problem (replacing the X Window system). And as a result of this they created their own replacement. If we were to compare this process to the reflexive standardization process which Hanseth et al describe in their case about the DocuLive ERP system, the most striking similarity is that when adding more actors to the process, the standard became less stable (1996). The more projects involved would cause more requirements to be discussed and in the end this adds up to the complexity of the process it self, but presumably also to the protocol definition, although this is not something I have looked at in this thesis. The defection of Ubuntu from the Wayland collaboration do however not add to the complexity to the Wayland project per se, but it adds to the complexity of the display server layer. I have in illustration 9 modified Hanseth et al.'s illustration to make it fit with my case.

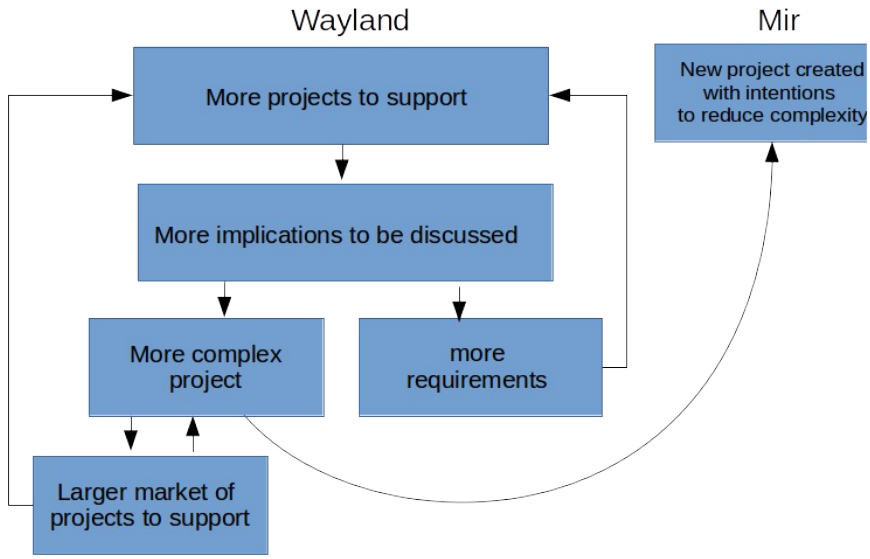


Illustration 9: Reflexive standardization Mir

As more actors join the Wayland project, more implications needs to be discussed, resulting in both more requirements that goes back into the project and a more complex project. And as a side effect of this, one actor decides to create its own with intentions to reduce the complexity (at least for themselves). The end result is a display server layer with more disordering, which can be viewed as a meta-change within the infrastructure (Beck et al., 2003). This also changes the social coordinates in which the display server layer previously were managed by the Free Software Community, which is what you would expect as a result of a second modernization or reflexive standardization (Hanseth et al., 2006). Going back to Pinch and Bijker's example of how social values forms artefacts, and being aware of that values change over time, the transformation of the working relations of those who maintain the systems also changes over time (Pinch and Bijker,1984;Cohn, 2013;Ferraro & O'Mahony, 2007) This type of change in both infrastructure and social relations could thus possibly be a reflection of a shift in values and principles in the Free Software community at large, and might be one of the consequences from the possible fact that Free Software is

becoming more used in commercialised environments (Fosfuri, Giarratana, & Luzzi, 2008;Dahlander & Magnusson, 2005).

6. Conclusion

I have in this thesis explored how organizations who are involved in the making of Free Software are connected, what considerations they make when deciding on technologies, and by looking at a controversy within a process of standardization studied how the process can be shaped by values and reflexive mechanisms. I made use of the notion of a recursive public in order to answer my first research question, which was to find out in what ways my actors were connected. What the answer to the first question revealed was that not only where they connected through the technologies which they were all using in their own projects, they were also connected through the users of their systems. This observation helped me understand not only how they were connected but also gave some answers to how their actions could cause controversy between the other involved parties. The technologies which they all use acts as intermediaries between these organizations and also affects the path in which the respective software projects evolve. By situating myself in the position of the actors which I interviewed and through the perspective of path dependence I also found that although their paths is to some extent determined by their previous choices of technology, there are ways for them to take control over this situation. Their path dependence was not as important as anticipated but the concept was useful in order to answer my second research question, which also led to a discussion on what kind of mandate the code represented when the actors made

considerations. The code seemed to have a strong mandate when my actors chosen what technologies they were to use in their projects, but what problems the code solved depended on what the respective actor viewed as relevant for their own projects. This finding supports Pinch & Bijker's model of how artefacts are being created by solving problems for the relevant social groups. For my case, this perspective help us to understand how software evolves over time (1984).

From a standardization perspective, the creation of Wayland as a multi-purpose display server seems to have caused an unintended amount of complexity, mostly for the collaboration process. This situation led one of the actors to create an alternative (Mir). This two folding of directions which the two alternative display systems adds to the overall complexity of display servers and can thus be explained as an end result of a reflexive standardization process, where the creation of Mir can be viewed on as an unintended consequence of the initial standardization effort. The creation of Mir also serves as a good example of how Free Software organizations creates software as arguments as opposed to debating in order to express themselves in how they think infrastructure should look like.

In retrospect, I also had to ask myself: what is there to be learned from this case and where do the thesis contribute to existing knowledge about standardization processes? The case provides insight about where you could expect friction when trying to balance commercial activities with community driven practices in a symbiotic like relationship. It also provides an example as to what could occur if standards are created which does not find the correct balance between what would enable actors to innovate on top of standards and what prevents them to do so.

Given the fact that software is still a relative new craft in our society, I think there is a lot to question in terms of how software is currently being implemented and used in various con-

figurations throughout our society today. At the same time, there is lot to learn from those who work on the edges, and by taking note from activities such as those described in this case, organizations in our society could leapfrog much of the errors and mistakes that these organizations are experiencing; errors and mistakes that come as a result of being pioneers in a new paradigm of how things are done.

7. Appendix

7.1 Questions used in the interviews.

The questions were used as a template during the interviews and not followed chronologically. They were also modified slightly to fit with the different roles of the interview objects.

- Fist, could you just start introducing yourself and tell me something about your role in Canonical and a little about what working in your position mean?
- How does Canonical monitor, analyse and evaluate upstream projects? And how much does the experience vary when working with the different upstream projects.
- Could you say something about why the X Window system is insufficient in terms of delivering the Ubuntu experience.
- Maybe you could say something about why you decided to jump off Wayland and start on your own replacement to X.
- Are there any other actors are involved in the making of Mir display server apart from Canonical?
- How much do you pay attention to the responses from the community or others during the development of Mir?
- Do you see any other organizations in the future utilizing Mir in their own projects in the future?
- Is it reasonable to draw a parallel to a case of standardization ownership, similar to the UNIX wars in the 80/90s?
- Could you say something about what kind of critique that have been addressed towards canonical in the Mir/Wayland flame war.
- Are your team trying to avoid such comments in order to stay focused on your work or does it affect your motivation(both ways)?
- Are you aware of any ethos within the FLOSS community that fosters less fragmentation between the different projects?
- What do you prefer the most in FLOSS in general, diversity or standardization?
- Do you know of any actor you consider more important than others when it comes to the future of either Mir or Wayland or Xorg? That I maybe should contact?

8. References

Ancell, R. (2014, March 24). Why the display server doesn't matter. Retrieved from http://bobthegnome.blogspot.de/2014/03/why-display-server-doesnt-matter.html

Baldwin, C. Y., & Clark, K. B. (2006). The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model. *Management Science*. Retrieved from http://www.jstor.org/stable/20110584

Beck, U., Bonss, W., & Christoh, L. (2003). The Theory of Reflexive Modernization. Retrieved from http://tcs.sagepub.com/content/20/2/1.full.pdf

Bergquis, M., & Ljungberg, J. (2001). The power of gifts: organizing social relationships in open source communities. *Blackwell Science Ltd*. Retrieved from http://www.idi.ntnu.no/grupper/su/courses/tdt10/curricula/P2-4-bergquist01.pdf

Bijker, W. E. (1997). *Of Bicycles, Bakelites, and Bulbs - Toward a Theory of Sociotechnical Change*. MIT Press. Retrieved from http://www.google.no/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CB0QFjAA&url=http%3A%2F%2Fcstpr.colorado.edu%2Fstudents%2Fenvs_5110%2Fbijker.pdf&ei=5A4KVJPNDYXIyAOZooDQBg&usg=AFQjCNHWPZHLFI4HiKb8jX8-VQdJkvHu-g&sig2=-Qd8PiCea6V4ckszfSiPXA&bvm=bv.74649129,d.bGQ

Brown, E. (2013, March 6). Canonical's Windowing Shift: More than a Mir Techie Footnote. Retrieved from http://www.linux.com/news/embedded-mobile/mobile-linux/707710-canonicals-windowing-shift-more-than-a-mir-techie-footnote

Byfield, B. (2014). KDE Desktop vs. GNOME Apps: The Great Paradox. Retrieved from http://www.datamation.com/open-source/kde-desktop-vs.-gnome-apps-the-great-paradox-1.html

Callon, M. (1991). *A sociology of monsters: Essays on power, technology*. Retrieved from http://www.unc.edu/~jbecks/comps/pdf/callon.pdf

Chesbrough, H. (2005). *Open Innovation: A new paradigm for understanding indusrial Innovation*. Oxford University press.

Che, X., & Lewis, D. (2010). IPv6: Current Deployment and Migration Status. *International Journal of Research and Reviews in Computer Science (URRCS)*. Retrieved from Current Deployment and Migration Status

Clark, D. (1992). Proceedings of the Twenty-Fourth Internet Engineering Task Force. MIT, Cambridge. Retrieved from http://www.ietf.org/old/2009/proceedings/prior29/IETF24.pdf

Cohn, M. L. (2013). Lifetimes and Legacies: Temporalities of Sociotechnical Change in a Long-Lived system. *University of California, Irvine*.

Collins, H., & Pinch, T. (1993). *The Golem: what everyone should know about science*. Cambridge University press. Retrieved from http://cstpr.colorado.edu/students/envs_5110/collins_the_golem.pdf

Colman, G. (2004). The Political Agnosticism of Free and Open Source Software and the Inadvertent Politics of Contrast. *University of Chicago*. Retrieved from https://evols.library.manoa.hawaii.edu/bitstream/handle/10524/1583/09_pdfsam_aq_cultures_opensources.pdf?sequence=1

Cox, M. J. (2014). *About the OpenSSL Project*. Retrieved from http://www.openssl.org/about/

Dahlander, L., & Magnusson, M. G. (2005). Relationships between open source software companies and communities: Observations from Nordic firms. *Science Direct*

Research Policy. Retrieved from http://www.idi.ntnu.no/grupper/su/bibliography/pdf/OpenSource/Dahlander2005.pdf

Dalziel, H. (n.d.). Oh no, not Mir problems for Ubuntu? Retrieved from http://www.concise-courses.com/security/mir-problems/

Davis, P. (1985). Clio and the Economics of QWERTY. *American Economic Association*. Retrieved from http://www.jstor.org/stable/pdfplus/1805621.pdf?acceptTC=true

Edwards, P. N. (2002). Infrastructure and Modernity: Force, Time, and Social Organization in the History of Sociotechnical Systems. Retrieved from http://pne.people.si.umich.edu/PDF/twente.pdf

Edwards, P. N., Jackson, S., Bowker, G., & Knobel, C. (2007). Understanding Infrastructure: Dynamics, Tensions, and Design. Retrieved from http://cohesion.rice.edu/Conferences/Hewlett/emplibrary/UI_Final_Report.pdf

English, B. (2008). *Are projects hosted on Sourceforge.net Representative of the population of Free/Open Source Softare Projects?*. Retrieved from http://www.umass.edu/opensource/schweik/documents/Population_of_OSS_projects.pdf

Fedoraproject. (2014). Fedora Engineering Steering Committee. Retrieved from https://fedoraproject.org/wiki/Fedora_Engineering_Steering_Committee

Fedorawiki, F. (2014). Staying close to upstream projects. Retrieved from http://fedoraproject.org/wiki/Staying_close_to_upstream_projects

Ferraro, F., & O'Mahony, S. (2007). The Emergence of Governance in an Open Source Community. Retrieved from http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CCwQFjAA&url=http%3A%2F%2Fwww.techforce.com.br%2Fnews%2Fcontent%2Fdownload%2F18046%2F70638%2Ffile%2FOMahonyFerraro2007AMJ.pdf&ei=mOIFU53fDsHOtQbswYGABg&usg=AFQj

CNGeuEmYYN89XB4xvuHMxrFZL5uWTA&sig2=DHpvOoLgqabg_I6jEyXREA&bvm=bv.61725948,d.Yms

Fosfuri, A., Giarratana, M. S., & Luzzi, A. (2008). The Penguin Has Entered the Building: The Commercialization of Open Source Software Products. *Organization Science*. Retrieved from http://www.jstor.org/stable/25146180?

Foundation, L. (2014, June 2). Linux Foundation Training Prepares the International Space Station for Linux Migration. Retrieved from http://training.linuxfoundation.org/why-our-linux-training/training-reviews/linux-foundation-training-prepares-the-international-space-station-for-linux-migration

Gajewska, H., Manasse, M. S., & McCormack, J. (1990). Why X Is not Our Ideal Window System. Retrieved from http://www.std.org/~msm/common/WhyX.pdf

Gartner. (2014). *Gartner Says Smartphone Sales Grew 46.5 Percent in Second Quarter of 2013 and Exceeded Feature Phone Sales for First Time*. Retrieved from http://www.gartner.com/newsroom/id/2573415

Gates, B. (2014). MS DOS v1.1 and v2.0 (Version v1.1). Retrieved from http://www.computerhistory.org/atchm/microsoft-research-license-agreement-msdos-v1-1-v2-0/

GENIVI. (2014). *About GENICI*. Retrieved from http://www.genivi.org/about-genivi

Ghosh, R. A. (2003). Clustering and Dependencies in Free/Open Source Software Development: Methodology and Tools. *IDEI/CEPR Workshop on "Open Source Software:*. Retrieved from http://dxm.org/papers/toulouse2/cluster-final.pdf

GNOME, W. (2014). Complete the GNOME Wayland port. Retrieved from https://wiki.gnome.org/ThreePointEleven/Features/WaylandSupport

Godfrey, M. W., & Tu, Q. (2000). Evolution in Open Source Softare: A Case Study. Retrieved from http://svn-plg.uwaterloo.ca/~migod/papers/2000/icsm00.pdf

Graesslin, M. (2014). Why the Display Server Does Matter [Blog]. Retrieved from http://blog.martin-graesslin.com/blog/2014/03/why-the-display-server-does-matter/

Gräßlin, M. (2014, May 7).

Greenwald, G. (2014). NSA Prism program taps in to user data of Apple, Google and others. Retrieved from http://www.alleanzaperinternet.it/wp-content/uploads/2013/06/guardian.pdf

Habermas, J. (1991). *The Structural Transformation of the public sphere - an inquiry into a category of Bourgeois Society*. MIT Press. Retrieved from http://www.google.dk/books?hl=en&lr=&id=e799caakIWoC&oi=fnd&pg=PR11&dq=+The+Structural+Transformation+of+the+Public+Sphere:+An+Inquiry+into+a+Category+of+Bourgeois+Society,+&ots=5OLJkZVWz4&sig=c24Wvw3M6YTZ3cwhPkd0akGMAUM&redir_esc=y#v=onepage&q=The%20Structural%20Transformation%20of%20the%20Public%20Sphere%3A%20An%20Inquiry%20into%20a%20Category%20of%20Bourgeois%20Society%2C&f=false

Haller, J. T. (2013). *Open Source License Popularity*. Retrieved from http://johnhaller.com/useful-stuff/open-source-license-popularity

Hanseth, O. (1998). Understanding Information Infrastructure. *Universitetet I Oslo, Senter for Informasjons Teknologi*. Retrieved from http://heim.ifi.uio.no/~oleha/Publications/bok.html

Hanseth, O., Jacicci, E., Grisot, M., & Aanestad, M. (2006). Reflexive Standardization: Side Effects and Complexity in standard making. Retrieved from http://heim.ifi.uio.no/~oleha/Publications/misqsi3979r2.pdf

Hanseth, O., & Lyytinen, K. (2010). Design theory for dynamic complexity in information infrastructures: the case of building internet. *Journal of Information Technology*.

Retrieved from http://www.palgrave-journals.com/jit/journal/v25/n1/pdf/jit200919a.pdf

Hanseth, O., Monteiro, E., & Hatling, M. (1996). Developing Information Infrastructure: The Tension Between Standardization and Flexibility. *SAGE Publications*. Retrieved from http://www.ics.uci.edu/~andre/informatics223s2009/hansethmonteirohatling.pdf

Hauben, M. (2010). The history of ARPA leading up to the ARPANET. *Columbia University*. Retrieved from http://pages.infinit.net/jbcoco/Arpa-Arpanet-Internet.pdf

Heen, T. F. (2001). A few observations about systemd. Retrieved from http://lwn.net/Articles/453009/

Hippel, E. von, & Krogh, G. von. (2006). The Promise of Research on Open Source Software. Retrieved from http://www.idi.ntnu.no/emner/tdt10/curricula/P5-3-krogh06.pdf

Høgsberg. (2012). [ANNOUNCE] Wayland and Weston 0.85.0 released. Retrieved from http://lists.freedesktop.org/archives/wayland-devel/2012-February/002072.html

Høgsberg, K. (2008). Wayland gets a terminal. Retrieved from http://hoegsberg.blogspot.no/2008/12/wayland-gets-terminal.html

Hughes, T. P. (1987). The Evolution of Large Technological Systems. *Book: The Social Construction of Technological Systems*. Retrieved from http://www.f.waseda.jp/sidoli/Hughes_1987.pdf

ISO/IEC, 7498-1:1994. (2014). *Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model*. Retrieved from http://www.iso.org/iso/catalogue_detail.htm?csnumber=20269

Jackson, A. (2010). Fedora representive discussing the move to Wayland on Fedora mailing list. Retrieved from https://lists.fedoraproject.org/pipermail/devel/2010-November/145273.html

Kelty, C. (2008). *Two Bits - The cultural significance of free software*. Duke University Press. Retrieved from http://twobits.net/pub/Kelty-TwoBits.pdf

Lakhani, K. R., & Wolf, R. G. (2005). Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Softare Projects. Retrieved from http://ocw.mit.edu/courses/sloan-school-of-management/15-352-managing-innovation-emerging-trends-spring-2005/readings/lakhaniwolf.pdf

Larabel, M. (2013). Ubuntu Announces Mir, A X.Org/Wayland Replacement. Retrieved from http://www.phoronix.com/scan.php?page=news_item&px=MTMxNzI

Larkin, B. (2013). The Politics and Poetics of Infrastructure. *Bamard College, Columbia University*. Retrieved from http://www.annualreviews.org/doi/abs/10.1146/annurev-anthro-092412-155522

Latour, B. (1993). *We have Neer Been Modern*. Harvard University Press.

Latour, B. (2005). *Reassembling the social*.

Latour, B., & Woolgar, S. (1986). *Laboratory Life: The Construction of Scientific Facts*. Princeton University Press. Retrieved from http://www.amazon.com/Laboratory-Life-Construction-Scientific-Facts/dp/069102832X

Lee, C. P., Dourish, P., & Mark, G. (2006). The Human Infrastructure of Cyperinfrastructure. *Proceedings of the ACM Conference on CSCW*. Retrieved from http://www.ics.uci.edu/~gmark/CSCW06.pdf

Lefebvre, C. (2014, April 10). Interview with founder of Linux Mint Clement Lefebvre.

Lessig, L. (2006). *Code version 2*. Basic Books. Retrieved from http://codev2.cc/download+remix/Lessig-Codev2.pdf

MIT, wiki. (2014). Athena history (1983 - present) from A to Z. Retrieved from http://web.mit.edu/acs/athena.html

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two Case Studies f Open Software Developmen: Apache and Mozilla. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.114.5989&rep=rep1&type=pdf

Moglen, E. plea for Free Software before the European Parliament (2013). Retrieved from http://www.europarl.europa.eu/ep-live/en/committees/video?event=20130709-1530-COMMITTEE-JURI

Musk, E. (2014, June 12). All our Patent are Belong to You. Retrieved from http://www.teslamotors.com/blog/all-our-patent-are-belong-you

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Retrieved from http://nakamotoinstitute.org/bitcoin/

Neff, G., & Stark, D. (2002). Permanently beta: Responsive organization in the internet era. Retrieved from http://academiccommons.columbia.edu/download/fedora_content/download/ac:129215/CONTENT/2002_05.pdf

Neuhaus, C., & Polze Andreas. (2014). Cloud Security Mechanisms. *Technische Berichte Nr. 87 Des Hasso-Plattner-Instituts Für Softwaresystemtechnik an Der Universität Potsdam*. Retrieved from http://opus.kobv.de/ubp/volltexte/2014/6816/pdf/tbhpi87.pdf

Ohloh.net. (2014). *X.Org Project Summary*. Retrieved from http://www.ohloh.net/p/x

Pinch, T. J., & Bijker, W. (1984). The Social Construction of Facts and Artefacts: or How the Sociology of Science and the Sociology of Technology might Benefit Each Other. *Social Studies of Science*. Retrieved from http://www.ihs.uw.edu.pl/wp-content/uploads/2012/10/The-Social-Construction-of-Facts-and-Artefacts.pdf

Ries, O. (2013, March 4). taking Unity to the next level. Retrieved from https://lists.ubuntu.com/archives/ubuntu-devel/2013-March/036776.html

Ries, O. (2014, April 11). Interview with Director of Engineering at Canonical on Google hangout.

Roberts, J. A., Hann, I.-H., & Slaughter. (2006). Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal

Study of the Apache Projects. *Management Science*. Retrieved from http://www.jstor.org/stable/20110575

Sarini, M., & Simone, C. (2002). Recursive articulation work in Ariadne: The alignment of meanings. Retrieved from http://www.google.no/books?hl=en&lr=&id=AtPrRVIKNEAC&oi=fnd&pg=PA191&dq=recursive+articulation+work+in+&ots=-FGYQTgLiG&sig=j7y_N6568PB5wz6pxCQQZu3JIRI&redir_esc=y#v=onepage&q=recursive%20articulation%20work%20in&f=false

Scacchi, W. (2004). Understanding Open Source Software Evolution. Retrieved from http://www.ics.uci.edu/~wscacchi/Papers/New/Understanding-OSS-Evolution.pdf

Schmidt, K., & Liam, B. (1992). Taking CSCW seriously. *Kluwer Academic Publishers*. Retrieved from http://download.springer.com/static/pdf/497/art%253A10.1007%252FBF00752449.pdf?auth66=1398974410_3641ee9514cc35a057b0d82bb1c369b2&ext=.pdf

Seigo, A. (2014, March 24). More on why the display server does matter. Retrieved from http://aseigo.blogspot.no/2014/03/more-on-why-display-server-does-matter.html

Shuttleworth, M. (2010). Mark Shuttleworth » Blog Archive » Unity on Wayland. Retrieved from http://www.markshuttleworth.com/archives/551

Stallman, R. (1996). *Free Software, Free Society - selected Essays of Richard M. Stallman*. Retrieved from http://www.google.dk/books?hl=en&lr=&id=UJlNAgAAQBAJ&oi=fnd&pg=PA1&dq=we+maintain+this+free+software&ots=bLxo7UyNhs&sig=T3co-bSkkZKJXj1KlRhgSFZYrgc&redir_esc=y#v=onepage&q=we%20maintain%20this%20free%20software&f=false

Star, S. L., & Strauss, A. (1999). Layers of Silence, Arenas of Voice: The ecology of Visible and Invisible work.

Theimer, M. M., Lantz, K. A., & Cheriton, D. R. (1985). Preemptable Remote Execution Facilities for the V-System. *Computer Science Department, Stanford University*. Retrieved from http://lass.cs.umass.edu/~shenoy/courses/spring05/readings/Theimer_vsystem.pdf

Tsing, A. L. (2000). Public Culture volume 12. Retrieved from http://muse.jhu.edu/journals/pc/summary/v012/12.1tsing.html

UbuntuWiki. (2014). LTS - Long term support. Retrieved from https://wiki.ubuntu.com/LTS

Ven, K., Vereist, J., & Mannaert, H. (2008). Shouldd you adopt Open Source Software? *IEEE Computer Society*. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4497765

Venturini, T. (2010). Diving in Magma - How to explore Controversies with Actor-Network Theory. Retrieved from http://spk.michael-flower.com/resources/DivingInMagma.pdf

Wayland, F. (2014). Wayland FAQ. Retrieved from http://wayland.freedesktop.org/faq.html#heading_toc_j_0

Weber, S. (2004). *The Success of Open Source*. Retrieved from http://brie.berkeley.edu/research/SW%20Ch.1%20Dec02.pdf

Wilson, C. (2013). xorg/driver/xf86-video-intel (Version 2.99). Retrieved from http://cgit.freedesktop.org/xorg/driver/xf86-video-intel/commit/?id=58a7611

X.org, A. (2014). *About the X.org Foundation*. Retrieved from http://www.x.org/wiki/XorgFoundation/

Yamauchi, Y., Yokozawa, M., Shinohara, T., & Isdhida, T. (2000). Collaboration with Lean Media: How Open-Source Software Succeeds. Retrieved from http://delivery.acm.org/10.1145/360000/359004/p329-yamauchi.pdf?ip=193.157.137.45&id=359004&acc=ACTIVE%20SERVICE&key=CDADA77FFDD8BE08.8BE0DFE7B528F835.4D4702B0C3E

38B35.4D4702B0C3E38B35&CFID=457014189&CFTOKEN=85374285&__acm__
=1399832228_4dd417e69fbadf3e078d905277e451e3