// Read Between the Lines {of Code}//

_

Ward Goes 2016

Read Between the Lines {of Code}

Perspectives on Free and Open Source Software Development in Perspective.

Ward Goes

Sutent Numer: 5623618

Submitted to the Department of *Cultural Anthropology* of *Utrecht University* as part of the requirements for the degree of Master of Science

Supervisor: Walter Faaij

August 1, 2016

Index

Preface

The previous seven months, in which I created the following thesis on free and open source software development have been demanding and energising at the same time. Even though this study taught me a great deal as a future anthropologist, I hope my efforts have resulted in a thesis that proves informative to different academic and non-academic audiences. At this point I would like to address two of them in particular.

First and foremost I thank those individuals who have helped me during the process of producing this master thesis. The individuals I have encountered in the field have given me energy and have made me part of their inspiring and thriving communities. I want to thank the individuals who agreed to speak to me, who were friendly and inviting. Specially I thank the informants who have un-selfishly invested time and effort in my study and who have taken me in tow and introduced me to their communities. Finally I want to thank those fellow students who have been my close advisors, my parents for supporting me intellectually, mentally and financially during my studies and a special thanks to Walter, for his energetic, encouraging and sharp tutoring.

Secondly, I want to briefly advocate seeing the university and the faculty of social sciences as a site for studying technology. I want to specifically shout out to students in cultural anthropology. I respect individual interests of students, thematically coherent courses, and I understand the appeal of traveling the world and leaping into what has traditionally been considered to be the culturally unknown. However, academics that study technology are exploring and expanding the academic front-line, since technology is increasingly becoming part of our and our informants' sociocultural lives. I hope students and academics thus recognise that technology is part of all anthropological research and address it accordingly. This train is leaving and future anthropologist need to get on it. Let us admit that and keep it in mind when going into the field.

Ward Goes, July 30, 2016

4 Read Between the Lines {of Code}

Abbreviations

BSD	–	Berkley Software Distribution
CMS	–	Content Management System
Dev	–	Developer
Devops	–	Developer/Operator
Devroom	–	Space for programming together.
DOS	–	Disk Operating System
FLOSS	–	Free/Libre and Open Source Software
FOSDEM	–	Free and Open Source Software Developers European Meeting
FOSS	–	Free and Open Source Software
FSF	–	Free Software Foundation
FSFE	–	Free Software Foundation Europe
GPL	–	General Public License
ICT	–	Information Communication Technology
IT	–	Information Technology
Mac OS X	–	Macintosh Operating system
MIT	–	Massachusetts Institute of Technology
MS DOS	–	Microsoft Disk Operating System
BSD	–	Berkeley Software Distribution
OS	–	Operating System
OSI	–	Open Source Initiative
OSS	–	Open Source Software
PC	–	Personal Computer

6 Read Between the Lines {of Code}

Introduction

Computer technologies have become an integrated part of everyday human life. They are now mediating a myriad of human interactions. Anthropological studies on computer technologies are vital since these technologies are inherently social and cultural. First, because individuals frequently engage in social interactions through such technologies. Second, such human interactions imbue these technologies with meaning and remake them into cultural artefacts. The following study engages in such social and cultural connotations of computer technologies. It will deal with the social and cultural forces that flow from a particular type of software development: Free and open source software (FLOSS)[1] development. Based on fifteen weeks of ethnographic fieldwork, conducted at the beginning of 2016, I will demonstrate how the individuals who participate in this type of software development reform existing methods of working, organising and thinking.

► Coding is central to software development, since 'code' is the collection of instructions that ensure the functioning of software. To the layman, code has the estranging quality of alienating a certain reality, while simultaneously bringing the functioning of that reality down to its mathematical, or consequential essence. The layman does not understand the code in itself. Yet possibly he, or she has an understanding of how the code provides a certain service. To the layman understanding code, or coding is not merely a matter of understanding how to program a particular piece of technology. Rather, such understanding demands insight into the thinking about how technology works and what it does.

I was a layman when I started studying FLOSS for this master thesis in October 2015. Now, ten months later, I still am. Even though I did take a few online courses on Javascript in the past, doing fieldwork pointed out to me that I am practically incapable of producing even one proper line of code. However, after having been amongst software developers

[1] In this thesis I refer to free and open source software as "FLOSS." This is an abbreviation for *Free/Libre and Open Source Software*. As included in most academic work, 'Libre' comprises the notion of freedom. 'Free' is originally not intended to signify 'free of charge,' but rather signifies 'freedom' and 'freedom of speech.'

for three and a half months during my time in the field, I understand code (and how to talk about it) in a very particular way. As one informant put it: “Although you might know how to work with a hammer and nails, that does not mean you understand how to build a cupboard.”[2] Reversely I learned to communicate with coders — and understand communication between coders — about how they build and construct the cupboard, while I do not understand how they use the hammer and nails to do so. Understanding interactions about technology in such a particular way has allowed me to do an ethnographic study within the realm of computer technologies.

According to Tim Ingold the anthropological study of technology has long been rather unproductive since anthropologist saw technology as a denominator of certain materiality, instead of an action, or skill-based phenomena. Ingold argues that by having “placed technology beyond the pale of culture and society, as a quasi-autonomous system of productive forces, the way was open for anthropologist, at least those of a “sociocultural” persuasion, to ignore it” (2001,19). Anthropologist have long reasoned that “[a]s climate is for meteorologists and ecology for ecologists, so technology is for engineers” (Ingold 2001,19). Schiffer brings technology into the sociocultural realm as he states that “the anthropology of technology encompasses a bundle of research questions about people-artefact interactions manifest in activities at various scales” (Schiffer 2001,3). Indeed technologies are both media and cultural objects at the same time (cf. Mazzarella 2004).

Based on my time in the field with software developers, I add that academics should also look at technology as potentially flowing from, towards and in between individuals. Studying technological practices, rather than the technologies themselves makes them inherently social and cultural. Hence, I think of the computer technologies and practices I have come across during my time in the field as performative and sociocultural interaction. This has been particularly informative since exchange, collaboration and deliberation take a vital place within FLOSS development. Therefore academics discuss FLOSS not merely in its technical capacity, but predominantly in its sociocultural and political capacity.

In line with this discourse I will demonstrate that FLOSS development should not only be understood as a performative sociocultural interaction, but also as a transformative sociocultural force. Indeed FLOSS development shapes and re-shapes accepted ideas on software development, and development in general. Still, I see as the core pursuit of the

[2] Informant with interview (nr. 28, May 20, 2016).

following thesis to reopen the discourse and understanding of how FLOSS development does so. I will demonstrate that the true transformative forces that flow from computer technologies are revealed when they are being understood through the diversity of their wider sociocultural context. First, allow me to briefly introduce FLOSS development.[3]

► Software *source code* is the set of computer instructions in a human-readable programming language. The source code of *proprietary software* is foreclosed and its copyright is owned and commercialised by one designated party. Conversely, the source code of FLOSS is publicly accessible and operable. Anyone is allowed to use, modify and (re)distribute the code and software, or parts of it. Moreover, FLOSS development in principle relies on unsalaried programming, which assures a part of all FLOSS projects to remain non commercialised. These principles of FLOSS development have proven to produce high quality software, of which some of the most prominent examples are perhaps Android,[4] Wordpress,[5] Gimp,[6] or Mozilla Firefox.[7]

In the existing scholarly body of work, FLOSS is understood within a social, cultural and even political context, since "beyond the FLOSS movement's technical discussions of programming and code there lies a more fundamental concern with the re-negotiation of the sociocultural and industrial relations of property, knowledge, and, ultimately, power" (Bradley 2005, 12). Indeed FLOSS development is not merely discussed in its technical capacity, "but as a philosophy, a politics, a critique, a sociocultural movement, a revolution, or even a "way of life"" (Kelty 2004, 499). This is partly due to the fact that FLOSS development has rich historical and ideological foundations. *Free software* was brought into being in 1983, when Richard Stallman (founding father of free software) initiated the GNU project.[8] This project set out to "respect the users' freedom" (Stallman 2015, 79). According to *Free Software Definition*[9] these freedoms are:

---

[3] For those unacquainted with FLOSS development, in need of a more elaborate introduction might refer to some previously conducted studies (Coleman 2013; Gosh 2005; Kelty 2005; Sullivan 2011) on FLOSS development.
[4] Android is a Google initiated open source operating system based on Linux, available for mobile and tablet. https://www.android.com Accessed: July 22, 2016.
[5] Wordpress is a free software content management system (CMS) for web development initiated in 2003. https://nl.wordpress.org/about/ Accessed: July 9, 2016.
[6] GNU Image Manipulation Program. https://www.gimp.org/about/ Accessed: July 9, 2016.
[7] Mozilla Firefox is an internet browser, developed in 1998. https://www.mozilla.org/nl/firefox/desktop/ Accessed: July 22, 2016.
[8] GNU operating system. https://www.gnu.org/home.nl.html Accessed: June 25, 2016.
[9] Free Software Definition. https://www.gnu.org/philosophy/free-sw.en.html Accessed: July 22, 2016.

> "The freedom to run the program as you wish, for any purpose (freedom 0). The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this. The freedom to redistribute copies so you can help your neighbour (freedom 2). The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this" (Stallman 2015, 3).

This free software ideology advocates equal access to software and source code and the freedom to both privately and commercially use and (re)distribute software. It is here that free software development is entangled with larger societal issues. It touches upon "a broader culturally familiar vision of freedom, free speech rights, and liberalism that harks back to constitutional ideals" (Coleman 2013, 2).

Fifteen years later, in 1998, a separate movement was established. It brought into being *Open source software*. In the *Open Source Definition*[10] ten essential criteria were presented by the Open Source Initiative (OSI).[11] Among these criteria were (re)distribution, source code, integrity, discrimination and licensing. Rather than emphasising the necessity of software freedom, open source underscores the developmental qualities of bypassing copyrights and patents. (Carver 2005, 448). Open source software is: "software that harnesses the power of distributed peer review and transparency of process. The promise of open source is higher quality, better reliability, greater flexibility, lower cost, and an end to predatory vendor lock-in."[12] One of its founding fathers Eric Raymond reassures the practical nature of open source software. According to him it will "triumph not because cooperation is morally right, or software "hoarding" is morally wrong, but simply because the commercial world cannot win an evolutionary arms race with open-source communities" (Raymond 1998).

'FLOSS' is thus a blanket term for different types of software. In academic discourse both the moral ideology of free software and the practical ideology of open source software

[10] https://opensource.org/osd-annotated Accessed: July 22, 2016.
[11] https://opensource.org Accessed: July 11, 2016.
[12] https://opensource.org/about Accessed: June 15, 2016.

development are not understood as colliding only with proprietary software development. They are also discussed as a critique on neoliberal society at large, since they undermine the hegemonic understanding of property- and copyright. In FLOSS development ideas and goods are not licensed to shield them from others, but conversely they are licensed to ensure they remain accessible and useable to others. For instance Weber (2004) writes that the success of open source reconfigures existing ideas on property, copying and business. Sullivan (2011) claims that by reaching out to a audience beyond merely technologists FLOSS has become a social advocacy movement. Accordingly FLOSS is generally positioned as a disruptive force. This is visible in titles of academic studies on FLOSS such as *Coding Freedom* (Coleman 2013), *Decoding Liberation* (Chopra, and Dexter 2007), *Freedom Imagined* (Leach, Nafus, Krieger 2009) and *Hacking Capitalism* (Söderberg 2008).

Ergo, FLOSS development is often academically interpreted as a critique on neoliberalism and its foundational principles of open market, competition and property. Yet, "FLOSS is not a political platform. It is a modality of software production which finds its production founded in communal forms of decision-making, intellectual labour, and product distribution" (Bradley 2005, 5). In other words, due to its political agnosticism (Coleman 2004) FLOSS development does not aim to politically engage in a broader sociocultural, or political discussion, but the practices of FLOSS development do have sociocultural and political connotations. It has come to stand for a weightier and grander idea within the discourse on technology, development and society. It is merely impossible to study FLOSS without considering these sociocultural and political connotations of FLOSS practices.

► However, I will show that FLOSS is an elusive phenomena and emphasising its relation to property and neoliberalism produces a reductionistic understanding of it. Ho (2005) has previously demonstrated in her study on Wall Street bankers that by placing such emphasis academics yet once again confirm the universal accommodation and hegemony of neoliberalism and reason about FLOSS through a neoliberal rational. Notwithstanding the fact that FLOSS indeed reconfigures ideas on property, depicting the practices with this single colour would not do justice to the immense palette of beliefs, practices and practitioners.

The academic discourse on FLOSS development should not be considered impartial in shaping the perception of FLOSS practices. Therefore I will not merely analyse practices of FLOSS development. Rather, I will try to analyse the interplay between the academic

discussion about such practices and the practices themselves, since they are complexly intertwined and thus cannot be studied as separate. Hence, this is a thesis not just about a particular type of software development. It is as much about the language with which, and frame in which it is being discussed. Seeing that anthropology pre-eminently aims for ethnographic understanding of practices and practitioners, I study *why* FLOSS developers participate in FLOSS and analyse their reasons in relation to the overarching conceptual discourse. I started my research with the following research question:

> *Research question* — How are individual motivations of programmers to participate in free and open source software development related to the production and attribution of critical liberal ideological values to free and open source software development at large?

As my research has turned out and as will become clear in this thesis, the above research question is part and parcel to and symptomatic of the issue I will address over the course of four chapters. Perhaps the question is a legitimate and logical one when considering the discourse. Still, FLOSS development is once more questioned based on its relation to property and neoliberalism. This leaves only little space for insights that go beyond this singular conception of the practice.

Therefore, I will build up two main arguments over the course of the four chapters of this thesis. First, to indicate the singular academic understanding of FLOSS I will demonstrate how this question in itself confirms certain accepted ideas about the sociocultural and political aspects of FLOSS development and how it produces a reductionistic and specific answer tied to ideas on property and neoliberalism. Secondly I will postulate that a more productive way to study FLOSS development is to recognise the diversity of practitioners, practices, ideas and understandings within the field of FLOSS and understand how this diversity illustrates the sociocultural transformative force that FLOSS potentially entails.

► Seeing that I will not only study the practices of FLOSS development, but make the discourse on these practices object of study at the same time, I will review the discourse as if extracted from it, while at the same time being right in the middle of it. In other words, I will be critical towards this discourse while simultaneously my perception is to certain

extent a product of it. I am not aloof from the existing ways of thinking about FLOSS development and I position myself by definition in relation to them. Still, I do not intend to present my conclusions as clear cut, all encompassing, or unbiased. There are many people I have not spoken, many meetings I did not attend, many hackerspaces I did not visit and so on. This thesis does not give an all-embracing overview of FLOSS, and I do not claim it does. It is an attempt to reflect on the position of the practices relative to their conceptual context without making these practices submissive to this context.

This thesis needs to be read and understood within a larger framework of academic and public discourse on FLOSS development, while to a certain extent staying clear from this discourse. It is meant to be understandable to anyone with an academic, or non-academic interest, coming from either a technological, or non-technological background. Hence, at times you might find this text too elaborate, at others too concise. At times you might find it to be too abstract, at others too actual. Furthermore, this text is not an introduction to FLOSS, nor will it go into specific technical aspects of software development, programming, or infrastructures through which this software is being developed. I hope to provide insight into the practical and conceptual field around FLOSS' technical practices.

► As mentioned earlier, I draw the conclusions presented in this thesis from fifteen weeks of ethnography conducted from the last weekend of January to the second week of May 2016. During this fieldwork I employed different methods. I observed and participated at about thirty-five meetings ranging from lectures, coding sessions, sprints, hackathons, or social events. Besides, I conducted a little over thirty semi-structured and structured interviews. During this same time I also conducted online research on web platforms such as GitHub[13] and Meetup,[14] communication applications such as IRC[15] and Slack[16] and on organisations, foundations, blogs and projects.

I set out to conduct the bulk of my research online. I planned to apply methods of online ethnography, take in consideration the anthropological reflections on the internet and mostly focus on interactions through online platforms, mailing lists and communication applications (see Boellstorff 2012; Miller and Slater 2000; Wilson and Peterson 2002). Yet,

---

[13] https://github.com Accessed: June 28, 2016.
[14] http://www.meetup.com Accessed: June 28, 2016.
[15] Internet Relay Chat. http://www.mirc.com/index.html Accessed: June 28, 2016.
[16] https://slack.com Accessed: June 28, 2016.

quickly it became apparent that my qualitative and ethnographic outset for this thesis would not be met with merely online observations. Therefore I pivoted towards both off- and online research, with emphasis on offline research. Still, I have been interacting with informants through many different communication platforms, which demanded time in terms of both understanding how to use the technologies myself and how these technologies are being used by informants. At the one hand these interactions were of organisational nature: networking, scheduling meetings, or interviews. At the other hand I used audiovisual communication platforms to conduct interviews with informants located elsewhere outside of the Netherlands.

For the most part my collected data resulted from offline ethnography. The majority of the interviews and all the meetings were offline, scheduled mostly in Amsterdam, or Utrecht (the Netherlands). The setup of these meetings varied from informal coding sessions, to curated evenings with scheduled talks, workshops and networking, to classes on particular pieces of software, or particular coding skills. This variety of events allowed me to understand different ways in which FLOSS development manifests itself. During these meetings I *participated* if possible, but mainly *hung around*, *observed* and started *informal conversations* (Sluka and Robben 2007). I took field notes, which I processed and integrated into the collection of my data afterwards through cross-referencing and data triangulation.

In this thesis I will use information and stories from informants whom agreed to talk to me after I had clarified my intentions. I worked with *informed consent* (DeWalt and DeWalt 2002) to make sure informants were aware of the reach and ramifications of the information they furnished me with. Still, in the field this proved difficult in more complex, or volatile conversation settings. In such cases I only use observations and insights for more general conclusions. In the anecdotes and results presented in this thesis were provided to me in confidentiality and I therefore systematically anonymise informants and use pseudonyms. All the informants are then treated alike and it does not inadvertently frame the insights they provided me with in terms of academic value and sensitivity.

It is impossible to fully anonymise informants, since they might remain recognisable to peers, colleagues, or others in their professional fields. Therefore decisions are taken in consultation with informants. Only rarely I encountered such sensitive cases, since most informants had no issue with being mentioned by name, or recognised otherwise. Several informants emphasised that such information should indeed be out in the open,

since “it is a matter of public record.”[17] Others agreed to speak to me if the results of my study are published under creative commons.[18] Moreover, I predominantly communicated in English due to the international diversity of my informants. Still, I spoke Dutch among dutchman. In this thesis Dutch citations are translated and presented in English.

► Throughout the four chapters of this thesis I will demonstrate a certain discontinuity between the practices of FLOSS and the way in which FLOSS is being discussed. I will try to move beyond the discourse of liberal, or anti-neoliberal connotations and illustrate the transformative force that flows from FLOSS development in a different way. Individually, each of these four chapters highlights this discontinuity between practices and discourse in a particular way. Over the course of these four chapters I do not necessarily build a consequential argument. Rather, observations from the four chapters will be pieced together in the conclusion so that they together demonstrate how certain values are being produced and attributed to practices of FLOSS through language and discourse.

The first two chapters introduce the field of FLOSS as I have encountered it. They give a general impression of what FLOSS is, who is involved in it and what it signifies to the different practices and practitioners. More importantly still, these chapters highlight a particular observation regarding the relation between the practices of and discourse on FLOSS development. The first chapter (*Beyond Code*) shows that studying FLOSS development is not merely a case of code, coding and coders. The field around the actual practices of FLOSS development allows for many different stakeholders and includes many contributions from actors who do not posses programming capabilities. The second chapter (*A Matter of Perspective*) highlights the hybridity and dynamics of FLOSS beliefs and practices. Here, I problematise the implied homogeneity of beliefs by demonstrating the internal diversity between the founding fathers of free software, open source software and actors in the field. In doing so the chapter highlights that in the discourse these two are seemingly being blended together in both the practical and conceptual understanding. Using ‘FLOSS’ as an umbrella term for a range of practices is significantly different from using ‘FLOSS’ to denominate a range of beliefs. The last of the two is posited to be problematic.

---

[17] Casual Conversation with programmer, January 31, 2016.

[18] Creative Commons License. https://creativecommons.org/licenses/?lang=en Accessed July 10, 2016.

The third and fourth chapter both more directly address the main research question and the discourse on FLOSS development as conceptually tied to property and neoliberalism. The third chapter (*Cradled in Code*) demonstrates how the actual practices of FLOSS programmers come to being and why coders participate in FLOSS development. The process of becoming a FLOSS programmer makes for the internalisation of particular ways of learning and working. I then propose to understand the internalisation of FLOSS learning and working through the concept of enculturation. The fourth chapter (*Rethinking a Discourse*) shows that the foundational liberal ideology as promoted by the founders of free software since 1983 has led academics to built a firm sociocultural and political frame for understanding free and open source software. Yet, the conceptualisations in the academic discourse that conform to such a frame leave only limited space for other scholarly interpretations. This results in a singular and reductionistic understanding of FLOSS. Through the example of one of the most fundamental of FLOSS principles, the publicly accessible source code, I will illustrate how practitioners in the field interpret FLOSS principles differently.

The conclusion brings together these four discontinuities by both critically examining the main question and demonstrating how the configuration of the discourse incites the asking of such questions. Instead, I will propose to study FLOSS through all its abundant practices, practitioners and ideas. This will produce an understanding of FLOSS in practice. Besides I will show that this diversity specifically indicates the transformative force of FLOSS, as it branches out onto many different aspects of individual and communal lives and produces certain new ways of learning, working and even thinking.

Chapter 1
{Beyond Code}

► FLOSS is coded by programmers. Programming thus embodies the very core of FLOSS development and the values anchored in it. Anthropologist have studied how "[h]ackers[19] bring these values into being through an astounding range of social and technical practices" (Coleman 2013, 3). Especially the Debian Linux distribution has frequently been object of such academic research. Licensed under the GNU Public License (GPL), the Debian project is known to adhere free software values.[20] It "boasts an intricate hybrid political system, a developer IRC, a formalized membership entry procedure (the NMP), and a set of charters that includes the Constitution,[21] Social Contract,[22] and Debian Free Software Guidelines[23] (DFSG)" (Coleman 2013, 127). Studies on "hacker ethics" (Coleman 2005) in Debian, the distribution of work among Debian maintainers (Gregorio Robles 2005), or the organisation of Debian developers (Mateos-Garcia and Steinmueller 2008) allow for specific conclusions about Debian and its programmers. This makes Debian a rich and informative object of study.

Debian is merely one example of how FLOSS development is predominantly studied through programmers that partake in it. In line, contributions to FLOSS are seen as code based. At the start of my fieldwork I too set out to study FLOSS development through the practices of its programmers and the production of code. However, the following chapter will demonstrate that in order to understand how FLOSS practices are framed in a more general sense, academics should look beyond code and understand the act of FLOSS programming within a field of varying actors, activities and contributions.

---

[19] In contrast to its meaning in public debate, 'hacking' does not refer to illegal computer practices. Rather, in academic discourse a 'hack' is considered to be a solution to a technical issue, or bug. In academia programmers are called programmer, hacker, geek, coder and developer. I will refer to them as either programmers, or coders.
[20] Debian free software statement. https://www.debian.org/intro/free Accessed: July 11, 2016.
[21] Debian constitution. https://www.debian.org/devel/constitution Accessed: July 11, 2016.
[22] Debian social contract. https://www.debian.org/social_contract Accessed: July 11, 2016.
[23] Debian guidelines. https://www.debian.org/social_contract#guidelines Accessed: July 11, 2016.

1.1 Contributions

▶ At one of the last days of June 2013, even before I started my studies in Cultural Anthropology at the University of Utrecht, my father and me were on our way to Brussels. We were moving my belongings since I was starting a three month internship at a Brussels based graphic design studio. The car was packed. In Antwerp we made a quick stop to fill the car up with gas, which led us away from the highway into the suburbs of the city. Even though my father did not very often take along hitchhikers, at one particular crossroads we picked up Ralph, a middle-aged Dutchman who was living in France. Ralph had visited his family in the Netherlands and was hitchhiking through Belgium to return to his hometown Lille. My father proposed to take him along until Brussels. I offered him my seat in the front and squeezed myself into the backseat, in between my mattress, books and guitars. I could only partially hear the conversation between the two men in the front seat.

The conversation mostly covered Ralph's passion: Drupal,[24] a CMS (Content Management System) with which clients could easily manage their own web content. At that point I only had a vague and most likely skewed understanding of what Drupal is and what it does. I understood Ralph was involved with web development for which he used this particular piece of software. Although I had learnt about the web development platform Wordpress[25] by that time, I had never heard of Drupal before. I was fascinated with Ralph's story and his passionate, almost evangelical way of speaking about the Drupal community, the collaborative development and his contributions to this development. Yet, the difficulty I experienced to follow the conversation from the backseat and the excitement I felt due to my move to Brussels, made my mind drift away from the conversation. Ralph and my father kept their conversation going until we arrived in Brussels. There they exchanged contact details as we dropped him off at the exit at Brussels North. We said our goodbyes.

At the beginning of 2016 when starting up my fieldwork on FLOSS development, I was looking to get in touch with programmers involved with FLOSS projects. After days of loitering the Internet, lingering around in IRC chats and scanning e-mail lists for relevant topics, I suddenly thought of Ralph and the conversation he and my father had on the way

[24] www.drupal.org/about Accessed: July 11, 2016.
[25] www.wordpress.org Accessed: July 11, 2016.

to Brussels almost three years earlier. By then I had learnt about Drupal as open source software and I realised Ralph could thus be of value to my research. After acquiring his contact information from my father's address book, I e-mailed him. After one hour he answered with a lengthy mail, referring to blogs, websites and other writings he had been working on. That evening we had the first of three elaborate conversations on Drupal, his activities, writing, our shared passion for cycling and FLOSS.

► Having heard only parts of the conversation Ralph had with my father on our way to Brussels while relying on previous studies on FLOSS, I assumed that he was a software programmer himself. It became clear in Ralph his first e-mail that he was not. For the larger part of his life he had worked as a system administrator for a large American cigarette company which had its European headquarters based in the Netherlands. Afterwards he worked as a teacher in IT education for several years. He thus had a fair amount of experience and interest in computers and software. It was only when he was dismissed from this position in 2006 his path crossed open source software:

> "Well, at that point I became unemployed and I had to start searching for a job. I thought to myself: I need to publish my CV online. Then I started searching a content management system to do this…"[26]

Ralph started exploring Drupal and he has been working with it ever since. Still, he was not able to contribute to its development with written code, nor was he able to propose patches for particular bugs he would come across. At large he was not able to understand in what way the software he was using and which he helped developing actually worked.

Yet, the fact that Ralph is not capable of coding did not restrain him in contributing to the Drupal project and community. He has co-organised Drupal events in the past and at the time of our interviews he was putting together a Drupal gathering in Lille. For this so called Drupal 'sprint' Ralph was managing the accommodation, schedule, internet connection and so on. In general these Drupal sprints were meant for Drupal fanatics and coders to meet up, discuss issues they encountered, learn about Drupal and socialise. Hence, while Ralph lacked particular coding skills which would have allowed him to work on

[26] Interview with event organiser, nr. 2, February 23, 2016.

the software itself, Ralph had found his own specific way to contribute to the Drupal community. More importantly still, in word he was unambiguous about his motivations to participate in such an open source community:

> "Well, within open source I choose for Drupal. You see, I use Skype very often, since you cannot install open source software for everything. […] But I am interested in open source mainly because of the idea. Eh.. maybe a better world. That idea."[27]

Seeing that Ralph was merely one of the many cases of non-programming contributors I have encountered during my fieldwork, FLOSS development goes beyond merely code and coders. FLOSS houses a wide variety of prominent and less prominent actors who all contribute in different ways. In order to understand and productively discuss what FLOSS development signifies beyond its technical capacity, academics should consider a range of actors and actor groups among which FLOSS foundations, users groups, community builders, event organisers, activists and even businesses and academics. Only when scholars move away from a study of programmers onto the study of a field, they will capture how social and cultural meaning is produced and attributed to FLOSS. Below I will highlight stories of three of the actors I have met during my time in the field who are not coders themselves, but still facilitate their respective FLOSS communities in distinctive ways. In doing so, I further illustrate the fact that contributions are not necessarily about code.

1.2 Different Actors, Different Practices

► As mentioned above, using IRC chats and mailing lists to get in contact with potential informants did not seem to be very fruitful. I experienced the communication via these media to be of a technical nature. Besides, these media were not used to plan offline FLOSS gatherings. In search for informants I could get in touch with and for meetings I could attend, I moved my focus to specific social media: GitHub[28] and Meetup.[29]

---

[27] Interview with event organiser, nr. 2, February 23, 2016.
[28] www.Github.com Accessed: July 11, 2016.
[29] www.meetup.com Accessed: July 11, 2016.

GitHub provides programmers with the ability to communally develop software since it uses Git[30], a version control system, originally developed for the Linux kernel. FLOSS programmers share their open source code on GitHub to be able to work on it together with their peers. This incidentally allowed me to study certain projects and contributors online and assess if they could be of value to my research.

Still, Meetup proved to be more valuable. First of all since it is not built around code, or FLOSS projects, but rather around communities. Meetup made visible and tangible that FLOSS development is made up of different actors and different practices.[31] Secondly, the features that meetup.com provided me with led to a breakthrough in my research. It is an online platform that enables its users to start specific groups and organise meetings for group members (see appendix I). It is used for a wide variety of sportive, religious, cultural, educational and political activities. Still, it is mostly used for tech, or business based practices. The fact that Meetup requires all users to make a profile allowed me to very easily identify relevant groups and individuals who are at the very heart of certain communities (See appendix I). Ironically — and different from my expectations — these individuals were in most cases not programmers.

Jesper — In general FLOSS foundations such as the Free Software Foundation Europe[32] (FSFE), the Free Software Foundation[33] (FSF) and the Open Source Initiative (OSI) take a prominent role within the field and the discourse on FLOSS. Since its launch in 2001 the FSFE considers that "the central component of [its] work is keeping the legal, political and social base of Free Software strong, secure and free of particular interests."[34] At one FLOSS event I came across the FSFE booth. Even though I had heard of its existence and I had conducted some online research on the organisation, I had not spent much time on studying foundations within the field of FLOSS in general. I took a flyer and started reading. After e-mailing back and forth with one very helpful intern I got in touch with Jesper. A couple weeks later in mid April we had a conversation over the phone.

Jesper is co-founder, executive director and former vice-president of the FSFE. He has a background both in software engineering and mathematics. During our conversation we

---

[30] Git, version control system. https://git-scm.com Accessed: July 11, 2016.
[31] Fieldwork Observation (February - May) — online research on meetup.com.
[32] https://fsfe.org/index.en.html Accessed: July 11, 2016.
[33] http://www.fsf.org Accessed: July 11, 2016.
[34] https://fsfe.org/about/principles.en.html Accessed: July 11, 2016.

talked about his activities as a programmer briefly, but quickly moved on to the FSFE and Jesper's role within this organisation. To Jesper his involvement in the Free Software Foundation Europe serves as a mere retribution for all FLOSS had given him: "I wanted to be part of this community. I wanted to actually make a change to the way that we work with software, but a large part of it was also felt that wanting to give something back to this community that had given me so much previously."[35] In this case Jesper thus chose his role in the FSFE over his role as a programmer since he felt he could 'give back' through this platform. Indeed, software foundations have become an active part of the FLOSS landscape and actively shape the mission of and communication regarding FLOSS: "Our overarching goal is to put people in control."[36] This shows that the forces that flow from FLOSS are not necessarily coming from programming in itself. Moreover, it urges to think about FLOSS not only in terms of software, but also in terms of communities.

Peter — On a Wednesday night in March I attended a small Joomla[37] gathering in a office space in Bussum. About ten Joomla fanatics were there. Alike Drupal and Wordpress, Joomla is a CMS and web development platform used by web-designers to develop complex websites. After the scheduled presentation the others gave feedback to the speaker and discussed the contents of the presentation. It was about ten o'clock when I prepared to leave for the train station of Bussum to catch the train to Utrecht. I was about to say good-bye to the organisers and thank the speaker for his presentation when somebody noticed: "Aren't you going to Utrecht? Peter is going to Utrecht by Car. I'm sure he could give you a ride." Thirty minutes later Peter dropped me off at Utrecht central station after a comfortable journey and an interesting conversation.

Two weeks after, when I visited his studio space in Utrecht he explained he had been working as an independent web developer for about six years now: "I started with small clients, a lot of freelancers, small businesses. I still have those clients, but now I also work for larger companies. SME's [Small and medium-sized enterprises] and such."[38] Peter exclusively worked with Joomla, was active within the Joomla community in the Netherlands and wrote a book on search engine optimisation targeting Joomla development.

---

[35] Interview with free software advocate, nr. 19, April 15, 2016.
[36] Interview with free software advocate, nr. 19, April 15, 2016.
[37] Joomla is a free and open source content management system. The Joomla project was initiated in 2005, as it was forked from another FLOSS CMS project: Mambo. https://www.joomla.org Accessed: July 10, 2016.
[38] Interview with web developer, nr. 15, April 11, 2016.

Even though he is not a software developer and he cannot contribute to the development of Joomla in terms of code, Peter explained he still contributes to its development and community. Once every few months he spends one day testing new features and versions of Joomla at a so called "Pizza, Bugs and Fun" meeting. These meetings are designed to test new Joomla features, or versions and are attended by twenty to thirty people. "Whether you are a developer, translator, a programmer, integrator, designer, or anyone who occasionally works with Joomla websites is fit to attend."[39] On such testing days Peter tries to find bugs in new features and versions of the software, or talks with Joomla developers about improving the software. "Most people who attend such a 'Pizza, Bugs and Fun,' well, you get a slice of pizza at the end of the day, but you'll have to enjoy it in order to do these things." He emphasises that "they all help out voluntarily. […] Still, it needs to yield some sort of fulfilment. […] It is OK to do unsalaried work"[40] Since FLOSS projects often do not have the resources to do controlled beta-testing they often rely on FLOSS contributors willing to invest time in Joomla.

Massimo — A day after I sent him a message on meetup.com, Massimo, a FLOSS event organiser and entrepreneur, replied to my message: "I'll be glad to talk to you and help you in your research, I always wanted to be a guinea pig :)."[41] It showed the humorous and helpful attitude of the Amsterdam based Italian. He organised several gatherings in Amsterdam among which a monthly 'devops'[42] meet-up and a bimonthly openstack[43] users group. Furthermore he could boast on a great deal of experience since he has worked for Redhat[44] in the past, after which he started as an independent development and operator consultant and occasional recruiter within the FLOSS sector in Amsterdam. I decided to get in touch with Massimo because as an entrepreneur, organiser and expert he has a large network in the tech and software industry of Amsterdam and beyond. A week after our first online correspondence we sat down together in a noisy café in the centre of Amsterdam, and had a two hour discussion on FLOSS.

---

[39] http://www.joomlacommunity.eu/nieuws/joomla-in-nederland/976-joomla-pizza-bugs-en-fun-nederland.html Accessed June 23 2016.

[40] Interview with web developer, nr. 15, April 11, 2016.

[41] Chat conversation on meetup.com (February 24, 2016).

[42] Community around the integration of developing, operating and testing of computer and server systems.

[43] https://www.openstack.org/software/ Accessed June 23, 2016.

[44] Redhat is a multinational, commercial open source software enterprise. Its core business comprises supporting businesses that use open source software produced by Redhat. https://www.redhat.com/en Accessed June 23, 2016.

Massimo told me about his professional activities, how he indeed had worked for Redhat in the past and the meet-ups he organised. He introduced me to many different organisations, websites, companies, events, pieces of software, projects, programmers and other individuals that might be relevant to my research. Quickly it became clear to me that Massimo had built a large network within the software sector in Amsterdam over the course of his career. This allowed him to relatively easily get things done. Whether this meant sponsoring for a particular event, arranging a space for an openstack meet-up, or finding suitable programmers for the projects he was working on. Still, it is rather hard to measurably determine what it was that Massimo contributes to the FLOSS communities since he is a jack of all trades. Perhaps the most defining contribution Massimo did for different Amsterdam FLOSS communities was facilitating these communities and building the essential networks that underpin them.

► Whether it is by being a FLOSS advocate, community builder, tester, or one of the myriad other FLOSS tasks that go beyond code, when considering the above mentioned individuals and their practices it is fair to say that without producing actual code, they contribute to the development of FLOSS nonetheless. Even if one would consider their contributions to be irrelevant, or less relevant than actual coding, still these actors are to be recognised in academic studies, since they play an active role within the FLOSS field and their practices, voices and investments should be considered as such.

1.3 A Climate for Development

► The above profiles together demonstrate three things. Firstly, they illustrate how the programmers in these practices are accompanied and facilitated by many different non-programmers and non-programming practices. It seems naive to regard such practitioners and practices as irrelevant and leave them out of the equation, since they harness similar values and foster FLOSS communities. In fact, due to many of those non-coding actors FLOSS communities thrive. Secondly, since they are part of FLOSS communities they also actively shape these communities and the way these communities are perceived. Whether that is explicitly in the case of Jesper and FLOSS foundations, or implicitly in the case of

Peter and the example of software testing. Thirdly, they demonstrate that certain ideas and values of FLOSS development branch out beyond FLOSS programmers.

Academics studying FLOSS communities should look beyond technological practices and understand how a range of different roles facilitate a climate, or vague orderliness that accommodates the development of FLOSS. For instance Bollier (2008) has written that:

> "Perhaps the most enduring contribution of the free software, free culture, and other "open movements" has been their invention of a new species of citizenship. Despite significant differences of philosophy and implementation, these commons share some basic values about access, use, and reuse of creative works and information. No matter their special passions, the commoners tend to be improvisational, resourceful, self-directed, collaborative, and committed to democratic ideals. They celebrate a diversity of aesthetics, viewpoints, and cultures" (Bollier 2008, 364).

Bollier's use of citizenship is in this case problematic, since citizenship refers to societies. However, the above does capture how certain thinking echoes among groups of FLOSS practitioners, or 'commoners.' The similarity in their reasoning about the how and why of development brings these commoners together in the "severe effort of many converging wills" (Raymond 1999, 20). This is not to say that programmers do not claim a particular position within such communities, nor it is to deny that their practices are at the very core of FLOSS development. Rather, it is to say that when studying meaning making within FLOSS development it is productive to consider FLOSS development not as a detached, geeky and computer based set of practices, but as a set of practices firmly embedded in a community and a wider social and performative canvas.

Together these different actors thus build a climate for FLOSS development in which many different individuals saddle themselves with tasks of different size, prominence and impact. However different, these tasks ultimately all help further the development of software. My informants have referred to this symbiosis as a system, community, or ecosystem: "We compete on those things […] on which commercial competition is possible. For instance on services, or hours, […] but on the other hand we cooperate continuously, since we communally profit from it, as an ecosystem."[45]

[45] Interview with software developer nr. 25, May 6, 2016.

▶ *In Conclusion* — In this chapter I have demonstrated that studying FLOSS development as a community based process, climate, or ecosystem advocates for a new understanding of the making and re-making of meaning and values of FLOSS development. It then imbues a wide range of activities with the potential of meaning making. It as such considers many different actors and activities in the studies on FLOSS. Moreover, thinking about FLOSS development as a system, or ecosystem also creates an understanding of how customs of learning, working and thinking are reproduced and circulate through FLOSS communities. Whereas the discourse is grafted on the *act* of programming FLOSS, the range of practices affiliated to FLOSS development (see appendix II) that go beyond coding underscore the *communities* through and the *climate* in which FLOSS is being developed. Further on in this thesis it will become apparent that recognising FLOSS diversity and the reproduction of a climate for FLOSS development is essential.

Chapter 2: {A Matter of Perspective}

▶ The different FLOSS practitioners I described in chapter one were all to a certain extent familiar with and self-conscious about how their practices are being conceptualised. However this is partially because the ideological discourse on FLOSS development takes a prominent place within FLOSS communities, this awareness still signifies the inter-twinement of discourse and practices. Still, informants all had their own particular way of interpreting and explaining the relation between FLOSS ideology and their own FLOSS practices. There is not merely one way to see the relation between the two.

In this chapter I will demonstrate that incorporating these different perspectives in studies on FLOSS development shows that the notion of what FLOSS signifies is not at all homogeneous. In fact I will explain that meaning is produced and ascribed to FLOSS in many distinctive ways and it should thus be academically understood through such distinct perceptions. I will address the implied homogeneity that comes with the term 'free and open source software development' and demonstrate that its use is potentially out of place when employing it interchangeably to denominate both a range of practices and a range of ideas. The use of this term is symptomatic for the way in which FLOSS is currently being discussed in academic discourse.

2.1 A Different Truth

▶ FOSDEM[46] (Free and Open Source Software Developers' European Meeting) is one of the largest annual, non-commercial FLOSS conferences of Europe. At the end of January it takes place in Brussels. About 5000 FLOSS fanatics attended the 2016 edition with over

[46] FODEM homepage. https://fosdem.org/2016/ Accessed: July 22, 2016.

600 presentations. Judging from the names of most of the scheduled presentations — such as *Real-time Charging for distributed community platforms using CGRateS*, or *FPGA Manager & devicetree overlays*, or *Creating rich WebRTC applications with Kurento* — I was going to grasp only very little of most of the presentations that were going to be given over the course of this last weekend of January, which marked the start of my fieldwork. On Saturday, the first day of the conference, my international train arrived at Brussels Central Station a little after ten in the morning. The talk *Free as in freedom. The importance of FOSS in the surveillance era* by Giovanni Battista Gallus, scheduled at eleven, seemed to be comprehensible since it covered a side of FLOSS development I had been reading about for over three months. While I realised I still needed at least one hour to get to the Solbosch Campus, I understood I would not be able to make the only presentation that seemed understandable to me.

In the train I was trying to decipher yet another of the countless, cryptic presentation names in the Saturday schedule. I quite suddenly felt intimidated with the assignment I had given myself: studying practices I was utterly unfamiliar with. How was I going to interact with and draw conclusions about software developers without the technological knowhow? The nervousness I felt was perhaps understandable, yet in many ways ill-found. Over the course of my research, programmers have proven to be extremely helpful and patient with me. They would try to provide me with insight into their practices without in any way being patronising, or incomplete. Often programmers had practice, since they very regularly would try to explain their parents, families and friends what this computer thing they do actually is. Furthermore, most programmers take pride in what they code, build, fix, contribute, or develop and are very happy to talk about it. As it turned out, nor the lack of programming knowhow, nor the lack of experience with actual software development were problematic. Programmers understood I did not understand.

In fact, it has shown to be advantageous at times, since my incompetence allowed me to ask ‘stupid’ questions and converse about ‘simple stuff,’ or led coders to pull out their laptop and demonstrate. Still, the task of the informant to make his, or her ideas and practices comprehendible to the researcher is not to be considered natural by definition. It should not be ignored that the informants I interacted with during my time in the field were actively part of the translations made between informant and ethnographer. In his 1977 classic *Fieldwork in Marocco*, Paul Rabinow states that:

> "[T]he informant is asked in innumerable ways to think about particular aspects of his own world, and he must then learn to construct ways to present this newly focused-on object to someone who is outside his culture, who shares few of his assumptions, and whose purpose and procedures are opaque" (1977, 152).

Sometimes this process merely meant that my informants had to explain something three times over again for me to understand. At other times it meant they actively had to rethink and reformulate their translations. Their creative solutions and particular strategies with regards to translations did not just solely proved to be helpful in studying their practices. It also accommodated me with insight as to how these informants themselves reasoned about their practices, how they framed them and how they ascribed meaning to them.

► The FLOSS enthusiasts I met at FOSDEM were different in terms of background, field, programming language, interest, skills, nationality and ideas. The crowded cafeteria of the Solbosch Campus was filled with greetings of long forgotten friends, meetings of new acquaintances, heated discussions and above all: abundant laughter. Every now and then when the murmuring voices died down I would hear the buzzing and much subtler sound of numerous fingers hitting the keys of numerous laptops. As I enjoyed my sandwich in the corner I took in the colourful sights and sounds of the wide variety of individuals who harmoniously joined in on a three day "performance of a lifeworld" (Coleman 2010, 64). To FLOSS fanatics FOSDEM serves as the perfect occasion to flock together and socialise, code, learn, drink and game.[47]

During the first day I for instance attended a presentation on FLOSS licensing by an American copyright lawyer at the U building, I listened in on a friendly FLOSS developer elaborating about his FLOSS desktop project and I enjoyed a (to me overly technical) discussion on code compilers[48] in one of the devrooms[49] in the H building. The individuals I met, spoke to and observed were as much similar as they were different. They were as much together as they were apart. Indeed these FLOSS fanatics share certain ways of

---

[47] Ethnographic Observations (January 30, 2016).
[48] Computer program that translates written code in a programming language into machine code which is readable to, and operable by computers.
[49] 'Devroom' is an abbreviation for developer room. On mostly conferences separate rooms accommodate specific developers with spaces to discuss, explore and showcase software.

thinking (as I will demonstrate in chapter three) and they did in common spirit celebrate their unity. Still, the FOSDEM attendees seemed segregated. However FLOSS conferences are often seen to be a chance for birds of a feather to flock together, the unity during these celebrations should not be seen as fundamental. Even though they might be likeminded, they do not necessarily share a same notion of FLOSS.[50]

During the second day I engaged in many different casual conversations and conducted my first interview. I spoke to South Africa based programmer Roger who had come all the way to Brussels for the conference. Our conversation covered his business involvement with open source software, his history and track record within software development and the particular pieces of software he was working on at the time. At one point during the interview he explained he always experienced testing particular pieces of software on computers running on Windows to be a hassle, since "the licensing gets complicated. You can't just get a cloud machine[51] […] Ok assuming we are willing to pay for it, it's not a problem, but then how do you give access to multiple developers?" According to him testing on Linux is faster and simpler. While he was elaborating on this particular testing process he called himself a 'pragmatist.' Even though Roger seemed informed about, and to a certain extent concerned with the ideological side of FLOSS, he preferred FLOSS over proprietary software because:

> "[It] just has some features that are really really useful. Like if you get.. if you get software that's MIT licensed[52], or BSD,[53] since like GPL[54] is slightly more sophisticated, you don't need legal advice to install the software. You can use it at your company. You can distribute stuff. You don't care, […] It doesn't matter it's for free. It matters that it enables you to do certain things without overhead. Like, for me legal is always overhead."[55]

Since this was the very first interview I conducted, my world was slightly turned up-side down. Suddenly Roger introduced me to a entirely different truth. In academic literature I

---

[50] Ethnographic Observations (January 30, 2016).
[51] Virtual representation of the hardware of a computer.
[52] Massachusetts Institute of Technology License https://opensource.org/licenses/MIT Accessed: July 10, 2016.
[53] Berkeley Software Distribution License
[54] GNU General Public License http://www.gnu.org/licenses/gpl-3.0.html Accessed: July 10, 2016.
[55] Interview with programmer, nr. 1, January 31, 2016.

had studied meaning making in FLOSS development through property and neoliberalism. Roger showed me that FLOSS is understood differently by its practitioners. Of course I was already well aware that FLOSS development in practice is not per se tied to the free software ideology. Besides, as befits an anthropologist, I had expected to encounter a wide array of individuals, stories and truths. Still, this particular anecdote is one of the many examples that indicated there are very divers, even contradictory ways of both thinking about FLOSS practices and discourse. Both practically (chapter one) and conceptually FLOSS development is not merely one thing and it is questionable whether it should, or should not be addressed as such. Understanding FLOSS is not a comprehension of what FLOSS truly is. Rather it entails an understanding of how FLOSS is understood.

2.2 Beyond the Classical Standoff

▶ Exactly three months after that last weekend of January in Brussels, my Italian programmer friend Andrea and me were chatting over a beer on a Saturday afternoon at the waterside in the centre of Amsterdam. At one point Andrea pulled out his phone and proudly showed me a photograph of him and a slightly smaller, long haired and bearded man. On the photo Andrea seemed overjoyed. His face marked by a wide smile, his left hand placed on the other man's shoulder and his right held besides his face making a 'peace' sign. "Dont' you see who it is?" Andrea asked me. "It's Stallman!" Indeed, as Andrea called out his name I recognised Richard Matthew Stallman, software activist, free software advocate and founding father of the Free Software Movement[56], Free Software Foundation and GNU project. Andrea had attended FOSDEM the year before in 2015 and in one of the hallways of the Solbosch Campus he had spotted Stallman. He had walked up to his idol and boldly asked for a photo. By the time Andrea and me were chatting in Amsterdam I already knew Stallman had come to be a hero to many of my informants.[57] It was almost May and my fieldwork was coming to an end.

Two months earlier in the second week of March, I saw Stallman entering the stage of a conference room in Utrecht. A long round of applause escorted him. He set the tone of

[56] http://www.gnu.org/philosophy/free-software-intro.html Accessed: July 10, 2016.
[57] Interview with Programmer (nr. 9, March 10, 2016). Programmer (nr. 23, April 30, 2016).

his presentation by first asking for a three round hurray for NSA-whistleblower Edward Snowden: “Hurray, hurray. One more: Hurray.” The self-willed software protagonist had come to Utrecht to give a presentation on software privacy. Even though I did not know about the full extent of his cult-status at that point I read his essays on software freedom and understood the magnitude of his part in putting free software on the map. The presentation covered what I had already read in his writings and was therefore not startling. Still, during my fieldwork I became increasingly emphatic towards FLOSS development and the actors who participate in it. I too felt excited that Stallman was in fact standing there only a few steps away. Even though his initial ideology is perhaps less visible now, his status as figurehead of FLOSS has not been diminished.

In writing Stallman had described free software before as “software that respects users’ freedom and community. This means that the users have the freedom to run, copy, distribute, study, change and improve the software.” (Stallman 2002, 3) The movement that Stallman started in 1983 emphasises free software to be a call for freedom through and within technology. According to Stallman software freedom is important since “the nonfree program controls the users, and the developer controls the program; this makes the program an instrument of unjust power” (Stallman 2002, 3). Still, the free software ideology is merely one way in which to understand the significance of FLOSS.

As highlighted in the introduction, the largest conceptual difference within FLOSS is that between free software and open source software. The Open Source Initiative (OSI) profiles itself differently by emphasising the collaborative and functional aspects of FLOSS, allowing for a market oriented mindset. The principles of open source are described by American programmer and writer Eric Raymond, who is often seen as Stallman’s adversary. In his renowned *The Cathedral and the Bazaar* Raymond refers to open source software development as “a great babbling bazaar of differing agendas and approaches” (1999, 2). By using the analogy of the structures of both the cathedral and bazaar he compares the models of open source software development and proprietary software development.[58] These two models are visualised in figure 1. His treatment is especially informative when he compares the way in which both models deal with software bugs:

---

[58] Raymond initially used his analysis of cathedral and bazaar-like structures to address models of free software development. Only in the later released Halloween documents — a set of confidential Microsoft documents regarding the ‘threat’ of open source software that were leaked to Raymond — he addresses the difference between open source and proprietary software development with this same analogy.

> "In the cathedral-builder view of programming, bugs and development problems are tricky, insidious, deep phenomena. It takes months of scrutiny by a dedicated few to develop confidence that you've winkled them all out. Thus the long release intervals, and the inevitable disappointment when long-awaited releases are not perfect. In the bazaar view, on the other hand, you assume that bugs are generally shallow phenomena - or, at least, that they turn shallow pretty quick when exposed to a thousand eager co-developers pounding on every single new release. Accordingly you release often in order to get more corrections, and as a beneficial side effect you have less to lose if an occasional botch gets out the door" (Raymond 1999, 8).

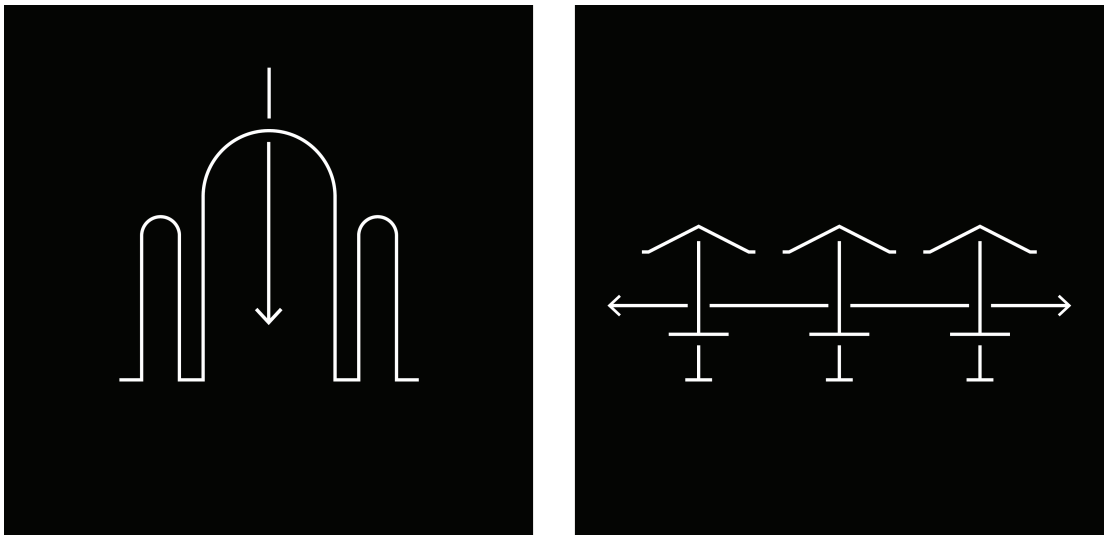


Figure 1. Models for software development by Raymond. The cathedral (left) and the bazaar (right). Image: Ward Goes.

As a reaction to Raymond's work, Stallman emphasised during his thirty minute presentation in Utrecht and in his essay *Why Open Source Misses the Point of Free Software* that: "[t]he philosophy of open source, with its purely practical values, impedes understanding of the deeper ideas of free software; it brings many people into our community, but does not teach them to defend it" (2002, 80).

Open source software does not cross swords with certain values of market and consumption, while "[t]he example set by free software (and a host of similar craftlike practices), however, should make us at least skeptical of the extent to which an ethic of consumption has colonized expressive individualism" (Coleman 2013, 14). Reasoned

through their respective foundations "[t]he FSF is more committed to provoking changes in software development by way of an appeal to the social use of information technology. The OSI is more concerned with implementing the technical practices that emerge from this activity within everyday computer use" (Bradley 2005, 2). In the light of the above, 'FLOSS' turns out to be an (in certain cases misused) umbrella term to academics. Bradley points out that it is "both an ideological movement *and* a practical methodology" (2005, 2).

'FLOSS' as a conceptual definition has come to stand for itself, whereas in fact it bundles together different, or even conflicting ideas. Also, the discussion on FLOSS is often outlined as the above classical conflict: Stallman versus Raymond, the FSF versus the OSI, or free software versus open source software. The definitions and notions that practitioners such as South Africa based FLOSS programmer Roger compose themselves are just as much part of a broader understanding of what FLOSS development signifies. Rodger for instance explained his particular view on such discussions:

> "I don't see so much of a distinction. I think it's more of a semantic distinction. You're calling it the same thing but depending on the contexts. Like, free software is a more politically motivated term than open source."[59]

Definitions thus move in between, beyond and even separately from the conceptual stand-off of Stallman and Raymond. I consider that a deeper and broader understanding of FLOSS is tied to the ethnographic understanding of practices and practitioners. In this broader notion, the term 'FLOSS development' does perhaps suffice in describing a collection of practices. Yet, I want to emphasise that it fails in academically describing a certain range of differing rationales. Regardless of whether they are rationales of FLOSS thinkers and writers, or practitioners.

2.3 Terminological Confusion

► As underlined by Stallman and free software advocates, the essence of free software is not to be understood within the context of charge, money and payment, but rather as tied to

[59] Interview with programmer, nr. 1, January 31, 2016.

freedom, free speech and privacy. An often used meme to describe this meaning of free is: 'free' as in 'free speech,' not as in 'free beer.'[60] Despite the fact that free software advocates emphasise 'freedom' rather than 'free of charge' does not mean free software is supposed to be understood as free of charge according to these advocates:

> "[W]e [the GNU project] encourage people who redistribute free software to charge as much as they wish or can. If a license does not permit users to make copies and sell them, it is a nonfree license. […] Since free software is not a matter of price, a low price doesn't make the software free, or even closer to free. So if you are redistributing copies of free software, you might as well charge a substantial fee and *make some money*. Redistributing free software is a good and legitimate activity; if you do it, you might as well make a profit from it."[61]

However the GNU project emphasises free software not to be 'a matter of price' — and even with the idea of what free software is and what it advocates firmly imprinted — the situation could seem somewhat contradictory to some, since notions on money and FLOSS differ. For instance "[b]usinesses and investors rarely understood the intended meaning and wrongly assumed that all free software had to be provided at no cost" (Carver 2005, 449). As an addition 'libre' therefore emphasises free software intentions.

For this study to grasp how free software (and FLOSS in general) is perceived by other actors active within, or outside FLOSS, it is informative to consider different ways of looking at the relation between free software and money. Not because either one of these perceptions is problematic, or just, but rather to recognise the diversity within the thinking about FLOSS. Here it also needs to be noted that whereas free software advocates indeed try to stand their ground on freedom, open source software advocates talk about open source methodologies increasingly as a business model. Any quick search on open source software in a random online search engine produces outcomes such as *Greed is good, Nine Open Source Secrets to Making Money*[62]*, How to make money from open source*

---

[60] https://www.gnu.org/philosophy/free-sw.en.html Accessed: July 17, 2016.
[61] https://www.gnu.org/philosophy/selling.en.html Accessed: July 31st. 2016.
[62] http://www.infoworld.com/article/2612393/open-source-software/greed-is-good--9-open-source-secrets-to-making-money.html Accessed: June 23, 2016.

software[63] and *How do Open Source Companies, Programmers Make Money*[64]. This does not mean that open source is thus a business model, but rather that it is being engaged with as such. Besides, the above indicates that the discussion on FLOSS is susceptible to communication and rhetorics. However the understanding of 'free' in free software as it was positioned by the founders of the free software ideology is the original, there are many different truths about what 'free' signifies with regards to software. The same goes for what 'open' means, or even what 'FLOSS' means.

At the end of February an Utrecht based FLOSS developer opened up to me about his childhood: "We were dirt poor, so I could not afford a lot of expensive tooling, and open source was free, and it gave me a way to get the tooling I needed."[65] He now runs his own software development venture in a studio space in the centre of Utrecht where he receives me. Furthermore, he is now a successful speaker at FLOSS conferences and organises a monthly FLOSS event in Utrecht. FLOSS has allowed him to elevate himself onto a financially stable life. In a similar way an open source developer from Poland explains that some do not afford proprietary software tooling, since:

> "Getting all the equipment to become an IOS developer costs a couple thousand dollars. You need a Mac machine, you need license for several things, and that for students, that creates a barrier that's too high […] So open source gives a very low barrier."[66]

Over the course of the lives of these particular informants FLOSS becomes an instrument of emancipation, since it allows the access to a vast body of instruments, information and other resources for free. This time free as in beer, not freedom.[67] FLOSS functions as an opportunity to self-teach and work oneself out of a social, or financial situation into another. Again, this does not mean that free software, open source software, or FLOSS should therefore be per see understood as such. Instead it means that scholars should take into consideration such notions.

---

[63] http://www.cio.com/article/2979583/open-source-tools/how-to-make-money-from-open-source-software.html Accessed: June 23, 2016.
[64] http://www.thewindowsclub.com/open-source-companies-programmers-make-money Accessed: June 23, 2016.
[65] Interview with programmer, nr. 3, February 26, 2016.
[66] Interview with programmer, nr. 1, January 31, 2016.
[67] Reference to "Free as in Freedom, not Beer." https://www.gnu.org/philosophy/free-sw.html Accessed: July 10, 2016.

- *In Conclusion* — A general conclusion can be made from the above examples regarding FLOSS as related, or unrelated to financial affairs. Regardless of the fact that ideological foundations of FLOSS do not take in views on such financial affairs, informants produce and ascribe meaning to FLOSS based on their own practices and experiences. The examples above indicate how informants themselves perceive the meaning and advantages of FLOSS, (or free software, or open source software) in relation to their own existence and subsistence. Despite the fact that practitioners might engage in somewhat the same technical practices, they think about those practices differently, within the context of their own lifeworlds. Often perspectives of practitioners move between, or even beyond the foundational discussion about and between free software and open source software camps.

Besides, unthinkingly using the term ‘FLOSS’ to indicate both practices and ideas leads to generalisation — of for instance different ideas on the relation between money and free software, or open source software, or FLOSS. This sheds an entirely different light on these practices and potentially offers a new way of studying FLOSS practices. The title of this section ‘Terminological Confusion’ therefore refers to the latter point made. ‘FLOSS’ as a term serves in indicating a range of practices, but it amiss binds together differing ideas and rationals at the same time. It is the collection of these different notions of FLOSS that indicates the width, reach and impact of FLOSS development as it branches out onto many individuals, groups, businesses, ideas and practices.

Chapter 3:
{Cradled in Code}

► The first and second chapter have highlighted that it is unproductive to consider FLOSS development as merely one thing with which only programmers are concerned. 'FLOSS development' is a placeholder for a wide array of practices and believes that complement, cross and contradict each other. Having said all this, it might seem paradoxical to devote the entire following chapter to the particular case of FLOSS programmers. Yet, the following study of programmers teaches valuable insights as to why individuals in general participate in FLOSS development. It serves as an illustration of the fact that thinking and talking about participation solely in terms of 'motivations' does not do justice to the role that FLOSS plays in lives of practitioners. In this chapter I will show that programmers do not participate in FLOSS development either because they find it fundamentally just, benefit from external, or future rewards, or feel competitive towards one another. I will demonstrate that they participate because FLOSS methods have simply come to be natural to them. All of these programmers are comfortable with FLOSS development since they have internalised certain ways of learning and methods of working.

3.1 The Making of a Coder

► "You need to destroy everything."[68] My informants explained that learning about computers is often equivalent to breaking them. Breaking a computer demands it to be repaired and repairing a computer gives insight into its functioning. Furthermore, in order to explore, destroy and repair computers future programmers need a fair amount of independency

[68] Interview with programmer, nr. 23, April 30, 2016.

and curiosity. It is one of the many signs that the way in which FLOSS programmers are educated is differently from conventional educational systems. This section demonstrates how this difference results in a different way of learning and working.

It should be noted that the vast majority of FLOSS programmers is male, and over the course of my research I have solely been in contact with male FLOSS programmers. Still, programmers are both female and male and this should thus not be overlooked. Especially since free software ideology and feminism are heavily intertwined and reinforced, and feminist technology studies have taken a prominent role in FLOSS studies (see Bray 2007). Besides, the below experiences and anecdotes of programmers need to be understood within a particular timeframe. Computing in the 90's was significantly different from what it is in the year 2016. In many ways computing for the general public was still in its infancy. Those individuals computing off the beaten paths had to be well rested with curiosity and perseverance. This is especially noteworthy seeing that the Internet was still far away from being web 2.0 and was not as user friendly as it is at this point. Information was only just becoming to be an easily accessible online good.

In this section I describe the childhood of a FLOSS programmer. However I present this programmer as one person, it is a compilation of the lives and experiences of different FLOSS programmers I interviewed. I incorporate bits and pieces from eight different interviews to outline a rich and complete profile of FLOSS programmers in general. These programmers I interviewed were all young men, in between 25 and 35 years old. They came from a wide variety of countries among which the Netherlands, Poland, the US, Italy, Romania, the UK and Colombia. Their particular stories all differ and their specific cultural and social backgrounds make each story particular. Still, from their stories common tendencies, interests and moments can be identified.

► Constantin's first encounter with a computer was in 1997 at the age ten. Constantin's interest in computers and technology developed over the course of the second half of the 90's as he started to experiment with the computer of his parents and the operating system that ran on it. Constantin was intrigued by the machine and he would constantly experiment by installing new pieces of software he wanted to work with. He would explore the operating system, or he would be surfing the web, scavenging for information. All of this often accompanied by a fair share of playing computer games.

> "I got into some really old DOS games and navigating in DOS and stuff like that and a new world opened to me. There were so many details I could go into and learn that there wouldn't be a very equivalent outside of this world. […] I never got into programming in that period. It was mostly learning on how it works, breaking it, reinstalling Windows, reinstalling DOS, […] getting basically into everything with the exception of programming."[69]

As Constantin explored the possibilities of computing he slowly but steadily pushed the boundaries and possibilities of his parents computer, which often led to heated discussions and friction: "On a regular basis my father had to reboot the computer, because I had caused Windows to loose it once again."[70]

Even though his parents were often driven to madness because of their son's whims, they harnessed and encouraged his technological curiosity at the same time. Constantin was using Dial-app, a service that offered phone based internet access during the 90. This was rather expensive, since such phone services were relatively new. During the night it would be cheaper, which caused him to spent entire nights behind his computer on the Internet. His mother would find him still hooked to the computer screen when she would get up early morning to get to work. She would scream out: "Come on, go to sleep, what the hell are you doing at this hour… you're still at the computer!?!"[71] Since Constantin had discovered the Internet to be a treasure trove of information and knowledge he had trouble restraining himself: "Sometimes I would stay during the day and I would just lose myself"[72] which led to exceptionally high telephone bills "and my parents would get pretty pissed off about it."[73] Constantin's parents then decided to avoid more exorbitant phone costs and had a permanent internet line installed which "was quite early. It was 2001 maybe, which for Romania was.. nobody had a permanent internet line."[74]

Indeed the Internet served Constantin very well in his development. In the 90's it was an unprecedented, almost magical wellspring of knowledge, which he exploited fully:

---

[69] Interview with programmer, nr. 29, May 26, 2016.
[70] Interview with programmer, nr. 20, April 19, 2016.
[71] Interview with programmer, nr. 29, May 26, 2016.
[72] Interview with programmer, nr. 29, May 26, 2016.
[73] Interview with programmer, nr. 29, May 26, 2016.
[74] Interview with programmer, nr. 29, May 26, 2016.

"It was my food you know. It was my soul food. I got a lot of information like that. I think it was one of the reasons why I was able to come here I would say. It opened a whole new world to me. […] I was going and just learning about everything online."[75] For instance blogposts, electronic magazines and software documentation made for "cool reading […] [on] how to exploit a computer, or a bug in the software, how to access a remote computer, how to run, or copy some files from one place to another, exploring the common line."[76] His friends would make fun of him because he would not get out of the house. The Internet "was the catalyst for everything basically" [77] he had already become very skilled and knowledgable in exploiting, managing and adjusting computers and software by the time he went to high school in 2002.

▶ Besides access to the Internet, there were two moments of upmost importance during his youth. The first was a little over three years earlier in 1998, when Constantin got his first (second hand) personal computer. He could still cite brand and model: IBM PC 330. The fact that he now had his own machine liberated him from the discussions and disputes with his parents. This also allowed him to take his computing and hacking a step further and continuously emerge in the world of computing. Moreover, this first computer allowed Constantin to build his identity around computation, since it functioned as the frame through which he perceived the world. Therefore this machine did not just bring about an array of possibilities, it also signified the establishment of a part of his identity as a programmer.

The second significant moment in the process of Constantin becoming a FLOSS programmer was the moment he heard about Mandrake Linux[78] for the first time in 2000. Constantin still remembered his friend had told him: "Wow, look at this operating system [Mandrake Linux], it's amazing. It never reboots, or you never get blue screens and it's so advanced. I know some guys who are hacking around. They are doing stuff with it and I think it's really cool."[79] Linux embodied Constantin's introduction to FLOSS software:

---

[75] Interview with programmer, nr. 29, May 26, 2016.

[76] Interview with programmer, nr. 23, April 30, 2016.

[77] Interview with programmer, nr. 29, May 26, 2016.

[78] Linux, or GNU/Linux is perhaps the most famous of FLOSS projects. Linux is a free operating system. Core element is the Linux Kernel that provides the communication between software and hardware by transposing the user input and output in data processing. Its project was initiated in the early 1990's and was initially built to work with GNU software, but by now works with more and more non GNU software as well.

[79] Interview with programmer, nr. 29, May 26, 2016.

> "[T]hrough Linux I basically learnt: "OK, it's open source, you can actually read the source about it. It was not that I was really going into the source, it was more the initial appeal. It was something new. It was used by more hardcore technical people and that attracted me."[80]

During the 90's Linux and other free software were new and exiting and indeed associated with more in-crowd hacking and advanced computing.[81] As Constantin started to use Mandrake Linux again innumerable possibilities opened up to him: "As soon as I heard about Linux I searched for it in Altavista[82] and figured out it was another operating system. It's a cheesy thing people say, [but] it is sort of a whole world of possibilities."[83]

Constantin experienced the adaptability, transparency and accessibility of Linux, which among others Windows, or Macintosh OS had never been able to offer him. Over the course of the 90's Linux had gained in popularity amongst likeminded computer fanatics, since the accessibility and adaptability of Linux distributions[84] allowed Constantin and his friends to faster explore, learn and hack together:

> "I went through them [distributions] […] frequently. One thing I really quickly appreciated was like.. there are so many different people doing so many different things, so you could just see visions on how to run an operating system, by installing one on your computer."[85]

Before he eventually started using OpenBSD, he had installed and tested dozens(!) of Linux distributions. Constantin became increasingly skilled in reading, modifying, or even writing code. The most advantageous feature of Linux in comparison to proprietary operating systems was perhaps the fact that he was suddenly able to both use a computer and at the same time look inside of it: "A new world appeared to me, because I started to sort of create

---

[80] Interview with programmer, nr. 29, May 26, 2016.
[81] Interview with programmer, nr. 29, May 26, 2016. Interview with programmer, nr. 23, April 30, 2016.
[82] Popular internet search engine in the second half of the 1990's.
[83] Interview with programmer, nr. 7, March 7, 2016.
[84] Software distributions are assemblages of a range of software parts, applications, libraries and so on. Linux, or Linux/GNU distributions (or distro's) are different collections of software that together make a Linux operating system and possibly offer additions to it. Examples include Debian, Ubuntu, OpenSUSE and Mandrake.
[85] Interview with programmer, nr. 7, March 7, 2016.

my own distribution, or modify everything, or understanding what I was doing. […] I really liked Linux and started to understand how it was working."[86]

► Besides the possibilities of the software itself, Linux led Constantin to become more and more involved with several FLOSS communities. This enabled him to access vast bodies of information and documentation on FLOSS software development. Furthermore, since he was starting to build relationships with FLOSS colleagues both online and at a Linux users groups in a nearby town, FLOSS became a part of his social life as well.

> "So I was introduced in a new type of IT let's say, where there was a community. People you can ask, people that were willing to answer your question, that were willing to help you understand something. This is why I was so exited to learn Linux, because before learning Linux I was at *my* home, using *my* computer."[87]

In 2005 Constantin switched completely FLOSS, which marked his full devotion to it and he became an increasingly integrated part of the FLOSS community, because of which he increasingly understood "what it all stands for and why it's important."[88] He became aware of the moral and practical value of sharing and the availability of resources and information, since he himself had benefitted so immensely from it. FLOSS communities helped him to become an independent and predominantly self-thought computer expert.

These FLOSS communities enabled him in learning about code and software, but he still had to muster a great deal of curiosity and perseverance in his search for information, code and as he was ploughing through documentation: "I was doing a lot of self study. Actually the default route: Try something, get stuck, consult Stackoverflow,[89] Google and so on."[90] He invested a great amount of time and energy in learning about programming and FLOSS. Still, he did benefit from it in diverse ways: "In certain subjects I was much better than the vast majority of my colleagues, because I already had a lot of experience. […] I had a good foundation."[90]

---

[86] Interview with programmer, nr. 23, April 30, 2016.
[87] Interview with programmer, nr. 23, April 30, 2016.
[88] Interview with programmer, nr. 29, May 26, 2016.
[89] Stackoverflow is a forum-like website that allows programmers to present each other with coding issues and help solve them online. http://stackoverflow.com Accessed: July 30, 2016.
[90] Interview with programmer, nr. 22, April 29, 2016.

Since Constantin was becoming an increasingly skilled programmer after the turn of the century, he became a more and more powerful contributor. Besides solely utilising the community as a resource, he was now able to contribute to this community at the same time. Contributing code to a FLOSS project for the first time was a weighty and insecure moment, because "code is really really personal. It's intimate in a way, because you're looking at the way somebody thinks about a problem,"[91] Some of his programming friends already contributed code at the age of sixteen, barely out of high school.[92] Others only started contributing code after twenty years of professional programming. Again others were are still getting to the point of contributing.[93] Constantin was a full grown and advanced developer even before he went to the university in 2006. He had been involved with OpenBSD already during his earlier teenage years. His commitment and dedication to the project resulted in him becoming the youngest OpenBSD developer ever:

> "My submissions were getting better and better, and with fewer feedback my submissions were committed[94] by another developer. At one point they were like: "Heck, just commit it yourself. It is of such size, and such quality.. It is better you do it yourself." […] I was eighteen when I got my account"[95]

Constantin had become a skilled FLOSS programmer. Now in 2016, being not only skilled, but also experienced, Constantin does not just execute a range of practices which are to be understood as FLOSS development. Through FLOSS he has firmly integrated a sense of independency and transparency in his thinking, talking, learning, working, building, communicating and documenting. Hence, learning to program through FLOSS has not just thought him to perform FLOSS practices, it has shaped his vision to a large extent. In the words of Constantin: "The true essence of the free software […] is something I apply on an everyday basis, on everything. For me it is a lifestyle because of this."[96]

---

[91] Interview with programmer, nr. 29, May 26, 2016.
[92] Interview with programmer, nr. 20, April 19, 2016.
[93] Interview with programmer, nr. 24, May 5, 2016.
[94] 'Committing' within the context of FLOSS is the process in which certain changes to the code, which are still under scrutiny and are in development, are approved, made definitive and permanently implemented in the software code by a programmer with the authority to do so.
[95] Interview with programmer, nr. 20, April 19, 2016.
[96] Interview with programmer, nr. 23, April 30, 2016.

3.2 Enculturation

▶ Many studies on FLOSS development study the 'motivations' of FLOSS programmers. (see Bezroukov 1999; Hars and Ou 2001; Hertel, Nieder and Hermann 2003; Lakhani and Wolf 2005; Lerner and Tirole 2001, 2002; Oreg and Nov 2007; Zeitlyn 2003). Lerner and Tirole (2001) for instance claim that programmers are motivated, since participation creates career opportunities, or wakes peer recognition. Hars and Ou (2002, 29) acknowledge such external and future rewards, but add to this what is referred to as "internal motivations" such as altruism and community identification. Bezroukov (1999) in turn poses that motivations of developers are triggered through the competition with proprietary software projects. Although these incentives are valid and I set out to study the motivations of FLOSS programmers as well, I have come across a more fundamental insight into why programmers partake in FLOSS development.

First and foremost, it must be recognised that solely speaking in terms of motivations would not do justice as to *why* programmers participate. Becoming a FLOSS programmer is in essence a process in which specific ways of thinking and working are internalised and are experienced as natural thereafter. This process is stimulated and facilitated by the FLOSS communities that programmers become acquainted with. I advocate an understanding of programmers' participation in FLOSS as linked to processes of enculturation.[97] At large, the concept of enculturation describes the process through which an individual learns about the accepted cultural and societal norms and values of his, or her environment. In this process a sense of boundaries and accepted behaviour is established Kottak (2006). Within academic work on FLOSS development the concept of enculturation is predominantly employed to describe the social and cultural process through which programmers become familiar with and adopt the communication, hierarchy, social structures, or best practices of particular FLOSS projects, or communities. Often this process is then referred to as a 'rite of passage' into a such a FLOSS project.

---

[97] Sociology and other disciplines within the social sciences use socialisation to denominate a similar concept. Often the two concepts are used interchangeably. Due to my background in cultural anthropology and my familiarity with the concept of culture as a dynamic set of practices, I will use enculturation from here on.

For instance Ducheneaut (2005) studies particular processes of enculturation in the Python[98] project. He states that "[i]t appeared clearly that being successfully integrated into an open source project is not as trivial as some would think: joining a project (and later evolving within it) requires one to go through a complex socialization process." (Ducheneaut 2005, 362). He uses the concept of enculturation to refer to the acquirement of a set of practices and behaviours in relation to others, rather than the internalisation of a general and intrinsic FLOSS value system. It describes the integration of programmers from a particular and specific sociocultural context into another. In a similar fashion Bach and Carroll (2009) describe enculturation as the process of understanding and bringing into practice certain values and practices: "because sharing practices with developers involves a process of enculturation: learning a rich set of moves and expectations, a variety of signals that members may not even be able to readily articulate but which they regularly and fluently enact" (2009, 239). Enculturation in this sense is informative when studying processes of socialisation in particular FLOSS projects, but it does not explain a more fundamental cognitive transformation of FLOSS programmers.

Coleman (2013) shows that FLOSS programmers internalise certain ethical and moral values. Indeed she thus refers to an internalisation of certain values. Still, in this case its concept is once again placed within the realm of the liberalist and anti-neoliberal ideology, since it is referred to as "ethical enculturation." Furthermore, it is still grafted on interaction between actors since ethical enculturation includes "learning the tacit and explicit knowledge (including technical, moral, or procedural knowledge) needed to effectively interact with other project members as well as acquiring trust, learning appropriate social behaviour, and establishing best practices" (Coleman 2013, 124). The concept of ethical enculturation is a valid one and Coleman's description of it does tip into a broader understanding of FLOSS enculturation. Nonetheless, understanding the internalisation of FLOSS ways of thinking and working in a broader sense demands it to be free from such ethical connotations, since the FLOSS process of enculturation I will describe in the next section goes far beyond processes of external socialisation.

[98] Python is an open source developed multi purpose programming language. The project was initiated in 1989 and was published under the OSI open source license. https://www.python.org Accessed June 28, 2016.

3.3 Ways of Learning and Methods of Working

▶ In response to the above notions of the concept of enculturation within FLOSS development, I propose a new use of the concept that traces back to the fundamental understanding of how FLOSS programmers internalise specific ways of learning and methods of working. As the case of Constantin has demonstrated, two particular aspects of FLOSS development are internalised during this process of enculturation. I will refer to them here as FLOSS learning and FLOSS working.

FLOSS Learning — All the FLOSS programmers I interviewed stated that they had a certain urge to explore: "In hindsight, it was what it was. It was just curiosity."[99] Curiosity seems to be a precondition rather than an convenient attribute, seeing that computing in the nineties was less straightforward: "It was weeks of slogging to get something working, especially with a CRT monitor. Incorrect settings could just blow up your monitor."[100] Besides being curious programmers thus need to be persevering and autonomous in solving the issues they encounter. While steadily obtaining the skills to overcome technical issues programmers also develop personal methods and structures[101] through which they tackle these issues. They then start to develop and internalise a sense of FLOSS development.

The introduction to FLOSS communities both saturates and enhances programmers' curiosity. The code, knowledge and documentation they get access to helps them to improve their skills, while it also visualises to them how much still is to be learnt and understood. The support systems that FLOSS communities offer could seemingly make FLOSS programmers become dependent on these communities. Yet, as they enter FLOSS communities their transformation into a programmer actuates into becoming a FLOSS programmer and contributor. This demands a higher level of independence in both learning and working. Regardless of whether FLOSS programmers drop out of high-school, or attain an academic degree, at an early age these young men already become self willed and skilled coders as they have educated themselves through the software, tools, information and documentation they have at their disposal. They learn how to learn, which constitutes FLOSS learning.

---

[99] Interview with programmer, nr. 7, March 7, 2016.
[100] Interview with programmer, nr. 20, April 19, 2016.
[101] Interview with programmer, nr. 20, April 19, 2016. On time management, workload, project structure, acquiring membership and programmer responsibilities within the OpenBSD project.

FLOSS Working — The fact that FLOSS developers are fit to function in this system is thanks to two reasons. Firstly, FLOSS working in fact relies on the autonomy required for the particular ways of learning discussed above. FLOSS development is often not facilitated with clear-cut structures, hierarchies and schedules and FLOSS programmers thus need to be independent to a large extent. They have to be able to make decisions, plan their own time and work. Simultaneously they need to be able to effectively communicate with fellow programmers, document their work and review other programmers' work.[102] FLOSS programmers do not have an organisational structure to fall back on. They need to be able to manage their own operations.

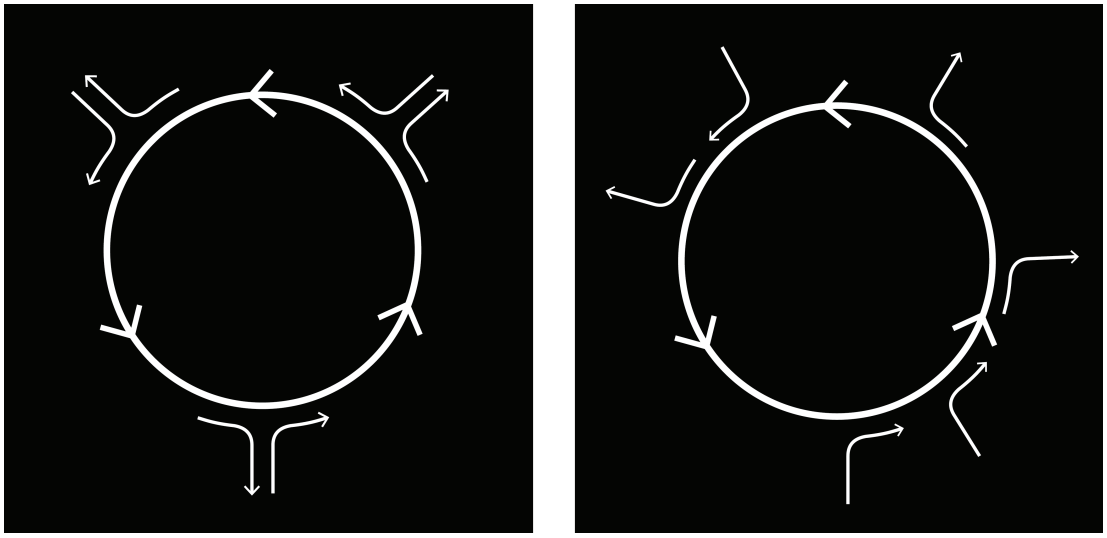


Figure 2. Direct and momentous transactions (left) and indirect and longitudinal transactions (right). Image: Ward Goes.

Secondly, as seen in the above mentioned anecdotes, it is during the process of exploring FLOSS that they become aware of the community and support systems. As they eagerly make use of them, they develop awareness about collaboration. By the time they work as full grown FLOSS programmers, contributing back to the community feels natural. On the one hand this comes from a feeling of owing the community. On the other hand it is simply the particular form of working they have gotten accustomed to. The combination of both moral and practical enculturation thus makes these programmers perceive transactions of information, knowledge and resources as indirect and longitudinal events, instead of

[102] Interview with programmer, nr. 20, April 19, 2016. on time management, workload, project structure, acquiring membership and programmer responsibilities within the OpenBSD project.

direct and momentous events (See figure 2). They have come to experience that it is beneficial to both themselves and a wider community to invest time and offer into FLOSS project and they have acquired an internalised sense of responsibility towards the the community. This perception grows from the feeling of "giving something *back* to the community" [Emphasis Added].[103]

- *In Conclusion* — Richard Stallman and the Free Software Movement set out in 1983 with a clear motivation of bringing to society's attention that software freedom is essential within an increasingly technological society. It has nonetheless (among others) resulted in the internalisation of certain ways of learning and methods of working that often do not originate from such ethically motivated ideas. FLOSS communities should accordingly be seen as 'communities of practice'[104] (see Grasseni 2004; Lave & Wenger 2002; Wenger 1998) that have a part in both external processes of social enculturation and the internal process of cognitive enculturation. Becoming a FLOSS developer is thus a natural process. It not a given choice, nor is it a matter of a series of events that evoke a decision. Being a FLOSS programmer stands for something more fundamental than participating in a series of certain practices on a regular basis. It entails these particular ways of learning and working and ultimately *being*. Reasoning about FLOSS development in line with the above could then produce studies that move beyond research on motivations of individuals towards understanding the internalisation and reproduction of such new and transformative methods.

[103] Informant during Coding Session, February 27, 2016. On contributions and contributing.

[104] What I propose here as being 'FLOSS learning' and 'FLOSS working' is to be studied within the context of separate social scientific debates on labour and skill. This thesis does not offer sufficient space to explore FLOSS development as related to such concepts and debates at this point. I will briefly address this potential line of inquiry in the section 'Discussion' on page 65.

Chapter 4: {Rethinking a Discourse}

► This chapter will place the ethnographic insights described above in academic perspective. I will show how the existing discourse on FLOSS development is constructed (see appendix III) and demonstrate it is to a certain extent singular and reductionistic when considering the diverse understandings of FLOSS I encountered in the field. I will first outline the academic discourse as it is now and show the conceptual implications of the enforcement of it. Then I will show how this academic understanding could be reconfigured and how it could offer space for more diverse interpretations of FLOSS. It is not the redefinition of property, property rights and neoliberalism that together indicate the true transformative force of FLOSS, but the collection of notions as found in the field.

4.1 In the Midst of Diversity

► From the tram stop at Dam square in Amsterdam I rushed to a lofty canal house at the Herengracht. I was late. This particular part of the Herengracht might not be as prestigious as the so called ‘Gouden Bocht’ (Gloden Bend) two hundred meters further down the canal. Still, the venue of that night’s event seemed extraordinary luxurious in comparison to other meetings I attended in basements, empty school buildings and industrial zones. I wondered whether I had mistakenly noted down the wrong address. Inside, when I climbed the marble staircase of the late 17th century building and smelled the scent of melted cheese and tomato sauce I was sure I was at the right place: Pizza! Five minutes later I was mingling in a crowd of FLOSS fanatics in the stunning central ballroom. The slice of pizza in my right hand was too cold, the beer in my left was too warm. Indeed I was late. It seemed like a paradox: eating pizza with my hands and drinking beer from the bottle in such a grand

and ornamented hall in which prominent tradesman would use to dine, dance and receive their guests during the Dutch 'Golden' Age.

Regardless of the venue, or budget of such meet-ups, pizza and beer are a tradition at most FLOSS events. They are typically served before the scheduled workshops, demos, or presentations, since these events are invariably held in the evening on weekdays, after office hours. The attendees thus come straight from work and instead of having dinner at home they have a slice of pizza at the meet-up. 'Pizza and beer' has become a notion in itself. It indicates an occasion for bringing into practice some of the most typical aspects of FLOSS development: socialisation, discussion and exchange. During such "pizza and beer" sessions I encounter the wide range of individuals and ideas that I described in earlier chapters of this thesis. 'Pizza and beer' thus signifies the mixing and mingling of all these different people, activities and opinions.

'Pizza and beer' is one of many traditions, customs and moments that made me see and understand that the notions of FLOSS development in general are not at all homogeneous. However they might not do and think the same, FLOSS practitioners are not put off by this diversity and complexity. Still, their communal complexity makes that FLOSS development becomes an increasingly elusive phenomena. This is perhaps one of the most valuable insights I have taken from my time in the field. Many evenings of eating pizza and drinking beer have made me see that an ever widening range of FLOSS practices, practitioners and ideas is being misrepresented by the current academic discourse in which FLOSS is mainly seen in the light of property rights and neoliberalism.[105]

In the scholarly discourse on FLOSS development the differently configured ideas and morals on progress and development are highlighted by perpetually placing FLOSS development in relation to property and copyright. Even though this reasoning is valid and it does provide conceptual insight, it is constructed through a very specific frame. Through this frame FLOSS is described as a highly technological form of liberalism and it is to certain extent positioned as a critical and opposing force to current neoliberalism. The first three chapters have emphasised the multiplicity of frames through which meaning is produced and attributed to FLOSS. These frames as found in the field prove useful in opening up the scholarly discourse on FLOSS and renegotiating academic understandings of actors, practices and ideas.

---

[105] Fieldwork observations (April 4, 2016).

4.2 Property and Neoliberalism

► The principles of ownership, property rights and copyrights are seen as fundamental within neoliberalism (Harvey 2005; Mansfield 2007). Moreover, these fundamental principles are also essential in the conceptualisation FLOSS and its relation to neoliberalism. If coding is an expressive activity, then some form of intellectual property rights and copyrights could be seen as corporate censorship (Kelty 2004, 503). Then the sole right to a creative idea is not merely a matter of property. It becomes a matter of free speech, since intellectual property rights and copyrights then constrain the freedom to express oneself (Bollier 2010; Coleman 2013; Kelty 2004). In other words, when a programmer is limited in expressing a creative idea through code because this idea is already patented, or owned by somebody else, this programmer is limited in his, or her expression and freedom of speech. In discussing FLOSS as a force that opposes such limitations academics bring it into the realm of anti-neoliberalism and activism (cf. Juris 2005, 2007; Juris and Razsa 2012; Nader 2011).

The current neoliberal project first occurred several decades ago, as it flowed from what has been referred to as Thatcherism and Reaganomics. Named after their respective introducers who had firm believe in such concepts, privatisation and deregulation since serve as the organisational fundament of structuring societies and exchange:

> "Whereas late capitalism is a *descriptive* or *explanatory* concept [emphasis added] that indexes a set of changes in the organizational structures of production and in relationships between states, industrial capital, and labor, neoliberalism is a *prescriptive* concept [emphasis added] that articulates a normative vision of the proper relationship between the state, capital, property, and individuals" (Ganti 2014, 92).

The development and the ubiquitous accommodation of this prescriptive concept have lead to globalised privatisation and globally accepted ideas on property rights (Harvey 2005; Hirsch 2010). Mansfield (2007) shows that privatisation and property are essential in neoliberal thinking: "The (neo)liberal rationale for property is based on this idea of complete control: the ability to use an object however one pleases, including the ability to exclude others from using it and the ability to transfer it to others (eg selling it)" (2007, 399).

The privatisation of expression and intellectual ownership might show significant resemblance to the enclosure of land in the period from the fifteenth to eighteenth century (Söderberg 2002). Yet, intellectual property rights manifest themselves less unambiguous. More specifically, in times of (both legal and illegal) digital copying and irrepressible global information flows, intellectual property rights seem to be further institutionalised in order to justify behavioural patterns of consumption. Reversely, these property rights and conditions of scarcity are also under pressure due to digital information and copying. Beebe explains that this causes an increasing implementation of such property and copyrights:

> "The emergence of fiat property, which is protected only because it is scarce, and scarce only because it is protected, predicts an emerging, though perhaps still distant, social role for intellectual property law. For all of its emphasis on "Progress," intellectual property law is emerging as a means to preserve certain conditions of scarcity and rarity that "Progress" is increasingly overcoming, and thereby to preserve social structures based on those conditions." (Beebe 2010, 888).

On the one hand technological innovation and progress are fundamentally overcoming the idea of property and on the other hand intellectual property rights are sustained to preserve certain conditions of scarcity regardless of technological innovation (Beebe 2010). Due to its specific methods FLOSS development bypasses intellectual property rights and copyrights and the underlying discussion on their functioning. In this capacity FLOSS development is seen as an act of reclaiming freedom through technology. "'Free speech' represents the freedom to use/modify/distribute the software as if the source code were actual speech which is protected by law in the US by the First Amendment" (Coleman 2013, 164).

► In order to give shape to the renunciation of copyright, copyleft[106] was brought into being in 1985 (see Kelty 2011) as the GNU project initiated the GPL (General Public Licence[107]). However this entailed a set of documented legal guidelines, the GPL was to reach beyond technicality: "Stallman intended software developers to use this license on their software

[106] https://copyleft.org Accessed: July 10, 2016.
[107] http://www.gnu.org/licenses/gpl-3.0.en.html Accessed: January 11, 2016.

and for software users to think about software freedoms. Contrary to the copyrights, copyleft licenses harness accessibility, modification and distribution of software" (Carver 2005, 455). Ever since, different copyleft licenses have been initiated. Among the more known are the above mentioned GPL, the Apache License[108] and the MIT License. In different ways copyleft licenses aim to regulate how particular pieces of FLOSS are to be used, modified and distributed. The GPL for instance declares that modifications made to GPL licensed software must per definition be publicly accessible and inspectable. In his 1985 GNU manifesto Stallman wrote:

> "GNU is not in the public domain. Everyone will be permitted to modify and redistribute GNU, but no distributor will be allowed to restrict its further redistribution. That is to say, proprietary modifications will not be allowed. I want to make sure that all versions of GNU remain free" (Stallman 1985, 99).

The MIT License does not have such demands. In turn it states that the developer of a derivative piece of software is obligated to mention the developer of the original software, or code. As such, MIT Licensed FLOSS software could potentially be used and modified for the development of proprietary software. Whereas copyleft originated from the free software movement in the early 80's, now it has been applied to many different types of information carriers such a for instance image, texts and so on. In spite of its establishment many years ago, copyleft remains under pressure due to corporate law and resistance from business sectors.[109]

Indeed the relationship between FLOSS and property- and copyrights (and thus neo-liberalism) is a strong one. Still, emphasising its relation to property rights and copyrights leads to a specific way of discussing FLOSS practices as a critique on neoliberalism. FLOSS might conceptually and practically re-shape thinking about property, ownership and development. The observations I have described in the previous three chapters have shown that FLOSS is a versatile, broad and elusive object of study. It seems that a new and broader academic perspective on FLOSS is necessary in order to understand its social and cultural implications in their full extent.

---

[108] http://www.apache.org/licenses/LICENSE-2.0 Accessed: July 10, 2016.
[109] 'Winning the Copyleft Fight' by Jonathan Corbet, 2016. https://lwn.net/Articles/675232/

4.3 Inevitability and Reductionism

► At large, this study on FLOSS development is yet another inquiry into emerging social and cultural formations resulting from processes of globalisation; in particular technology as a force of globalisation. It is an ever evolving process and a controversial and subversive subject within anthropological studies. Zygmunt Bauman describes globalisation as "'anonymous forces,' operating in the vast —foggy and slushy, impassable and untamable — 'no man's land,'stretching beyond the reach of the design-and-action capacity of anybody's in particular" (Bauman 1998, 60). This positions globalisation as a vague and out of reach process, beyond human power and understanding. Yet, globalisation is not just a mere phenomena, or given. Karen Ho states: "We need to look critically at globalisation as not simply a fact, but a hope, a strategy, and a triumphalist ideology" (2005, 86). In response to scholars attempting to capture globalisation in descriptive and defining systems (Appadurai 1990; Castells 2000; Eriksen 2007), others such as Tsing (2000) have advocated that globalisation is a project-based process, as it results from globally oriented projects pursued by individuals, groups and mankind as a whole.

Seeing globalisation as a project-based process makes it tangible and within reach, rather than just something out there. Besides, seeing globalisation as project-based connects its processes to actual human practitioners and practices. This is not to say that globalisation processes are influenced by individual actors, but rather that globalisation is a result of the conjuncture of certain manmade projects of different scale such as FLOSS development, or neoliberalism. Such projects of globalisation exercise certain cultural and social forces upon each other and consistently produce globalisation because of that. This is also not to say that projects of globalisation have apparent actors, univocal, or explicitly voiced goals, or consensual ideologies. It merely means that certain acts, decisions and processes together produce certain global forces. This view on globalisation does not aim for categorisation, or demarcation, but allows the retracing of power relations within and between such projects of globalisation and among the actors involved with such projects.

So then why is this analysis of globalisation relevant to this thesis? Frankly, FLOSS in itself is a highly technological project of globalisation, to which meaning is ascribed by FLOSS participants and collectives of different size. Through their practices these

practitioners frame FLOSS development and direct certain ways of thinking about it. Also the academic discourse on FLOSS leaves its mark on how it is understood. Since this discourse on FLOSS binds together practices, globalisation and neoliberalism, it confirms neoliberalism as a universal and hegemonic descriptive order. This while it tries to indicate FLOSS development as a potentially disruptive project of globalisation. It positions neoliberalism once again as the inevitable measure with which all other things are calibrated.

Anthropologists and other scholars have demonstrated that FLOSS is associated with liberal ideas and it is often seen as a critique on neoliberalism: “[F]ree software hackers not only reveal a long-standing tension within liberal legal rights but also offer a targeted critique of the neoliberal drive to make property out of almost anything, including software” (Coleman 2013, 4). Ho argues in her study on Wall Street bankers that hegemonic discourse and ideas on global capitalism are often enforced through scholarly studies since “[m]any cultural studies theorists and social scientists, by giving emphasis to capitalism's omnipotence, have helped to imagine a world of capitalist totality” (Ho 2005, 68). Academics might aim for the contrary: “In the rush to confront and depict the powerful impact of Western global hegemony, they have often neglected the power-laden political effects of their own representations of this very hegemony” (Ho 2005, 68). It becomes apparent that scholars are potentially complicit in creating powerful and leading projects of globalisation such as neoliberalism. “[I]t certainly would not appear to make sense for academics interested in counter hegemonic projects to paint the world using similar colours and tools” (Ho 2005, 70).

However the project of FLOSS is deemed to be an opposing, or critical force towards the neoliberal project, specifying FLOSS within the realm of neoliberalism colonises it and remakes it to enforce the neoliberal project. It paints FLOSS using neoliberal colours and tools. Discussing FLOSS in this particular way is paradoxical since it confirms the inevitability of universal neoliberalism. Both as an ideological philosophy (socialism and communism alike) and a project of globalisation, this inevitability makes us only very rarely realise that neoliberalism is merely one possible resolution to the fact that “[i]n all human societies there are conceptions of material and immaterial goods. The universal problem with which all societies have to cope is to regulate their members' relationships with respect to such goods” (Benda-Beckmann 1995, 310). This potentially creates a singular and reductionistic understanding of phenomena such as FLOSS development.

4.4 The Full Scope of FLOSS

The fact that the source code of FLOSS is freely accessible to anyone is perhaps the most fundamental of FLOSS values. Stallman's (2002, 43) definition provides: "Free software is a matter of the users' freedom to run, copy, distribute, study, change, and improve the software." Stallman presents the four fundamental freedoms that I have already mentioned in the introduction. In this case the second of these four freedoms is informative: "The freedom to study how the program works, and adapt it to your needs. (Access to the source code is a precondition for this)" (Stallman 2002, 43). This very fundamental and core idea of claiming freedom through software source code allows for the academic discussion of FLOSS in the realm of property rights and neoliberalism. In this section I will turn to this principle and demonstrate it is this discussion of FLOSS source code that illustrates a potential misconception. Bezroukov (1999) states that:

> "While it is true that the vast majority of users of Linux do not have the skill, nor the motivation, to add anything to the OS, other things being equal, the suitability of a program for a particular user is higher if the source code is available, as the source code is the ultimate documentation to the program. The possibility of making changes is less important especially for a large program. It is simplistic to assume that all open source users are C[110] programmers that are both capable and willing to make changes and/or correct errors in the OS and tools. Most users of Linux use it the same way they use Windows — they install the ready-made distribution and just apply patches until a new more attractive version arrives. Then the cycle is repeated" (Bezroukov 1999).

Most users do not have the slightest idea how to access, let alone adapt source code to their own whims. Most Linux users groups therefore merely focus on learning how to work with the OS.[111] One informant who organised Linux users group meetings jokingly

---

[110] General purpose and widely used programming language, published for the first time in 1973. C has branched into many different derivative C programming languages such as C++, Objective C, and Perl.
[111] Interview with users group organiser, nr. 23, April 7, 2016. Interview with users group board member, nr 18, April 14, 2016.

explained that these users group meetings are often ‘classes for elderly,’ since they often lapse into general courses on how to use computers in the first place. Another informant is looking to start up an OpenSUSE users group: “I just want to set up a group for support and guidance to use it.”[112] Even if users are capable of accessing the source code, the actual ability to understand and inspect it is an even bigger struggle, seeing that this is already an uphill battle for programmers:

> “Something that I find very false about open source: it is not true that developers check all the source. So it’s not that you use Linux […] you use open source and you can add your own changes to the operating system. You must be an extremely skilled developer. It will take years to understand how it works. And it will take a lot to change it. So, most of the time you don’t look at the code, but you know that somebody does. You know that some people like Stallman and his followers look into that and you expect it to be clean.”[113]

Regardless of the almost impossible task to work with the source code, the accessibility of the source code could still be perceived as valuable to the FLOSS users since it is “an additional form of consumer protection” (Bezroukov 1999). The above might very well fall under the same conceptual flag of what the value of accessible source code is in the light of property. In practice it indicates a variety of activities and ideas on what FLOSS is and what the value of free or open source code is. Some might see free accessible source code as a fundamental right, others might be comforted by the idea that it is inspected by skilled programmers, others again might know about it, but are not interested by the source code, but in the functionality of the software. There are many different notions of why FLOSS is potentially disruptive through its methods. Academic discourse often leaves those notions out of the picture.

► *In Conclusion* — A developer who grew up in an impoverished environment, with only limited future prospects might see FLOSS as instruments for emancipation;[114] a recruiter scouting skilled programmers at FLOSS meet-ups might see it as a helpful for doing her

[112] Interview with OpenSUSE user, nr. 14, April 8, 2016.
[113] Interview with programmer, nr. 9, March 10, 2016.
[114] Interview with programmer, speaker and event organiser, nr. 3, February 26, 2016.

job;[115] a Dutch programmer working in his spare time on an emulation of his very first MSX[116] computer from when he was twelve might see FLOSS as fundamentally enjoyable;[117] a South Africa based programmer might see FLOSS as a convenient way to avoid overhead;[118] a young anthropologist might see FLOSS as a colourful collection of beliefs and practices, which together are potentially disruptive precisely because of their diversity.

As a result from the anthropological holistic approach I had towards studying the practices of and discourse on FLOSS, I advocate understanding FLOSS its impact through the collection of all such divers and very subjective understandings of FLOSS by its practitioners. First, studies on FLOSS are then built on the contextualisations through which FLOSS practitioners themselves ascribe meaning to their practices. It gives insight into the actual meaning of FLOSS in its applied social and cultural contexts. Second, this understanding builds on the idea that the variety of such diverse contextualisations suggests that FLOSS development reaches into many different sociocultural and political aspects of society and human life. In this capacity it reveals its widespread force on individuals, communities, or society at large.

---

[115] Interview with tech-recruiter (e-mail), nr. 31, April 12 - May 2, 2016).
[116] First introduced industry standard for home computers in 1982.
[117] Interview with programmer (Google Hangouts), nr. 6, March 3, 2016.
[118] Interview with programmer, nr 1, January 30, 2016.

Conclusion: {In the Tangle of Reality}

► Over the course of four chapters I have highlighted different discontinuities between the practices of FLOSS development and the discourse around those practices. I will now return to the research question and show how these discontinuities shed new light on this question. I will demonstrate how the particular language used in this question confirms the same biased and reductionist premise I have been discussing over the course of this thesis. In its current form it thus produces a similarly reductionist answer. Despite the fact that some of these observation might seem obvious, or overly correct, together they eventually produce a different perspective on the way meaning is produced and attributed to FLOSS development. Subsequently they also result in a more productive answer and conclusion to this study as they urge to shrug off current academic connections with regards to FLOSS and break open its discourse.

► I have demonstrated that when academically discussing how FLOSS development practices are to be understood, it is productive to take into account a wider set of FLOSS practitioners rather than studying programmers. When studies take into consideration a wider spectrum of actors, activities and ideas they outline the climate, or ecosystem of FLOSS development in its widest sense, in which many different acts and actors contribute to FLOSS. This is not to say that studies on FLOSS programmers are not valuable. Rather, it is to say that scholars need to understand how all actors contribute to, or antagonise the production of a certain image of FLOSS. All these actors to different extent influence the general perception of FLOSS development and as such it is impossible to say something about programmers without placing them in the larger scheme of things. In other words, the research question of this thesis covers the relation between programmers and a larger discourse, but prior to drawing conclusions on how programmers relate to the discourse on liberal ideologies we

need to consider by whom and how this discourse is generally shaped and re-shaped. Whether that is by programmers, event organisers, evangelists, community builders, users groups, businesses, or foundations and so on.

Also I have demonstrated that the academic use of the term 'FLOSS development' is to a certain extent problematic. First of all since FLOSS as a conceptual field is not homogeneous. Using 'FLOSS' in an umbrella term to refer to a range of different practices is in fact valid. Yet, using the term as a conceptual tool in the academic discourse ties practices together as if they were conceptually similar. This reduces FLOSS and generates a singular contextualisation of the notions of for instance free software activist Richard Stallman, open source advocate Eric Raymond, or FLOSS practitioners whose ideas move in between and beyond those of Stallman and Raymond. These understandings are entangled in a colourful and multi-dimensional discussion and FLOSS as a conceptual term does not do them just. The use of the term 'FLOSS development' in the research question and academic discourse in general needs specification.

Programmers undergo a deeper and intrinsic process of enculturation that makes participating in FLOSS natural to programmers. The ways of learning and working which are very specific to FLOSS are internalised through years of experimenting, building, hacking, destroying and exchanging. It is inadequate to solely discuss why programmers participate in FLOSS development in terms of motivations, external incentives and future rewards. Instead, the process of enculturation makes the methodology of FLOSS resonate in programmers' practices by the time they become skilled programmers. It simply comes natural to them. This observation is derived from the particular case of programmers but leads to think that it is more productive scrutinising *why* FLOSS participants join in on FLOSS development in general, rather than using terms such as incentives and motivations.

Together these insights result in a reconfigured way of looking at the academic discourse on FLOSS. Currently it is shaped by academic reasoning about FLOSS, property and neoliberalism. Even though the conceptual understanding of FLOSS development through property- and copyrights is valid and should not be disregarded, I want to put forward that it is merely one. This understanding indicates only one way in which FLOSS development potentially proves to be transformative. I have demonstrated that FLOSS is for example transformative to some as an instrument for emancipation, as a process of socialisation and collaboration, or as a business opportunity. This should not be overlooked nor excluded

from the academic notion of it. When academics do not imbue FLOSS with one particular meaning this leaves space for appreciation of how FLOSS potentially grows, transforms and is being redefined. It frees FLOSS from the singular neoliberal contextualisation (cf. Karen Ho 2005). Understanding and positioning FLOSS as an opposing force to neoliberalism inherently makes it part of neoliberalism. It thus reconfirms neoliberalism as the benchmark in twenty-first century societies, even though academics often aim to indicate its erratic model. In other words, in their efforts to highlight FLOSS development as fundamentally undermining the role of property in society, their emphasis comes to reconfirm the role of property.

► Where does this leave us? What has the effort of this thesis of opening up the discourse and considering many different FLOSS practitioners, practices and ideas produced? It places academics right in the middle of a chaotic and dynamic field which is constantly developing and in which individuals enact a variety of beliefs and values in different ways. It leaves us right there in the tangle of reality. With this in mind I will now turn to an analysis of the initial research question.

> *Research question* — How are individual motivations (3) of programmers (1) to participate in free and open source software development (2) related to the production and attribution of critical liberal ideological values (4) to free and open source software development at large? [119]

The above observations do not only highlight problematic aspects of the current academic debate on FLOSS development. They also indicate problematic aspects which are part of the research question of this thesis. Also this question is solely focussed on programmers (1). It leaves unspecified the term 'FLOSS development' interchangeably in practical and conceptual (2). Its language confirms the ideas of motivations (3). Most importantly, it once again places FLOSS in the conceptual realm of liberalism and anti-neoliberalism (4). Therefore the research question has become an object of study in itself. Instead of answering the research question it is more constructive to analyse the answer it brings about.

---

[119] The numbers 1 to 4 refer to the different chapters of this master thesis, and are placed next to the terms in this research question to which the chapters refer and which they problematise.

To conclude this study it thus seems unproductive to answer a research question that I have experienced to be biased based on my time in the field with informants. Moreover, it is inconsistent to present the findings of this thesis in a pointy and trimmed answer. I could formulate an answer to this research question and let myself be limited by the very assumptions that confirm the premised discourse on FLOSS development. By using the concept of enculturation I could indicate that programmers experience a certain sense of naturalness to participation in FLOSS development. Therewith I could deny programmers' active allegiance to the liberalist and anti-neoliberal stance. However this would be a valid answer to such a research question, I want to stress this would again confirm hegemonic ways of thinking that aim to enforce, but eventually diminish FLOSS' transformative force. Denying the current academic approach that emphasises the relation between neoliberalism and FLOSS would be reductionistic and unproductive. First, because it confirms everything I have been arguing against over the course of this thesis. Second, because I then produce a deconstructive answer, rather than a (re)constructive perspective. Therefore I do not give a more elaborate answer to the research question, but instead propose a different approach to academic studies on FLOSS and end with a concrete conclusion nonetheless.

Alike the existing opinions, I too see FLOSS development as a potentially disruptive, or transformative force and FLOSS principles as potentially reconfiguring thinking about societal formation and configuration. Instead of regarding this force to result from the rejection of neoliberal thinking on property this study has shown that such transformative forces flow from the diversity of FLOSS practitioners, practices and ideas. Together they indicate how FLOSS principles resonate further than code, coders and software. The conceptual interpretation and practical appropriation of these principles by different FLOSS actors together imbue FLOSS with meaning. The principles, of which opening the source of products and the structures of knowledge are the most fundamental, thus branch out onto many different areas, even beyond technology,[120] for instance into democracy.[121]

If studies on FLOSS development incorporate such observations and retrace its impact through the range of individuals, activities and ideas academics uncover FLOSS' true force

---

[120] Evgeny Morizov states that in recent years "open" has become an increasingly popular term, used to denominate a wide range of practices and structures beyond technology, such as 'open democracies,' 'open communication.' http://www.nytimes.com/2013/03/17/opinion/sunday/morozov-open-and-closed.html Accessed: June 28, 2016.

[121] Douglas Rushkoff's treatment of open source democracy demonstrates how open source principles are potentially transposed toward other fields and structures.

on societies — whether those are neoliberal, or not. Studying what FLOSS signifies to its practitioners means studying how they learn and work. How they *perform* FLOSS, rather than how it is being *discussed*. This means scholars are to repeatedly connect and disconnect the different levels of this discussion. They are to consider the ethnographic understanding of the notions of FLOSS actors and practices and the ways in which academics conceptually frame FLOSS through specific idioms and reasoning as part of the same reality.

Admitting that FLOSS actors might not actively ascribe meaning to FLOSS, but rather feel it is 'normal,' or 'natural' is perhaps difficult to cultural anthropologists, since they who study culture *are* studying meaning making. Yet, it is essential to let go of certain ways of reasoning about FLOSS development and admit that its actors do not per se position their activities in relation to the larger scheme of things, since they don't perceive their own work as different, odd, or dissimilar to such larger schemes. While still highlighting and discussing more deeply the societal conceptual implications of FLOSS development, anthropologists should acknowledge that it is the tangle of reality in which practitioners thrive and in which FLOSS development reveals its true colours.

Discussion

▶ Based on the above I will now briefly outline certain ideas that could lead to a new course of further inquiry into FLOSS. I address the themes of diversity, discourse and exchange.

Diversity — This thesis has persistently emphasised the hybridity, heterogeneity and dynamics of FLOSS development. Studies that focus on the diversity of certain cultural and social formations often seemingly point out the obvious, since especially anthropology emphasises a dynamic understanding of cultural diversity. Besides, from a methodological perspective it leads to question if persistently describing such diversity is productive to academic activities in terms of formulating results and conclusions. Endlessly recognising diversity could lead to an academic impasse that would cramp academics and diminish their vigour and boldness as researchers.

I hope I have successfully shown that this potentially also works the other way around. Existing and hegemonic ways of academic thinking and working could just as easily result in cramped and incomplete academic work. The fact that academics have created a set of

conceptual understandings of FLOSS is not harmful in itself, as long as it serves them in perpetually reinventing the studies and discourse. Besides, recognising diversity is not only essential in order to portray FLOSS 'truthfully,' but to study it more productively as a force that is not merely engaged with technical phenomena, but a force that increasingly permeates society at large. Academics should take on the digital and transgressive chaos that postmodern times and globalisation have brought to society in the similar fashion in which they manifests itself: through meaning making in its most scattered and undefined manner. The meaning of FLOSS can then be reassembled and reinterpreted.

Discourse — After having denounced the discourse on property, copyrights and neo-liberalism as the dominant way to conceptually contextualise FLOSS development, I here advocate for other contextualisations. When the outset is to study a different type of citizenship (Bollier 2008), or a climate, or ecosystem for development, then it is essential to bring that study down to the fundamental traits with which actors together shape it. I propose to study the ways of FLOSS learning and working and discuss them within the discourse on labour and skill to understand what these concepts signify within FLOSS development. Marx for instance states that: "The external character of labour for the worker appears in the fact that it is not his own, but someone else's, that it does not belong to him, that in it he belongs, not to himself, but to another" (Marx and Engels 1978, 74).

Also FLOSS learning is to be integrated in a larger discourse on *skilled vision* and *communities of practice* (Grasseni 2004; Lave and Wenger 2002; Wenger 1998). In studies on FLOSS the latter concept means that "activities are specific to the community members and also share a tacit understanding of how to participate in the community" (Bach and Carrol 2009, 239). In itself this definition refers again to social enculturation, but in combination with the concept of 'skilled vision' these concepts together refer to the process in which actors shape their notion of the social and cultural context of their work, the particularities of that work that define the frame through which they see the world. The discourse on labour and skill serve as contexts in which to study FLOSS.

Exchange — Finally, there is one more way of looking at the issues I have outlined in this thesis that I want to briefly address here. This perspective does not per se propose a line of inquiry concerning FLOSS, but it is relevant nonetheless. Since I operated on both a conceptual and ethnographic level I have reviewed the interplay between two levels of inquiry. In doing so I have also addressed the negotiation between two groups and perhaps

also their methods: FLOSS practitioners and academics. On the one hand I demonstrated that the specific redefining methods and values of FLOSS development resound in practitioners and their practices through a lifelong process of enculturation. On the other hand, when reviewing the discourse on FLOSS, I have put forward that in academic studies on FLOSS there resounds a certain methodology of reconfirmation. It seems that also academics develop, find and sustain certain 'natural' ways of thinking, working and talking. The title of this conclusion (In the Tangle of Reality) therefore also refers to the reality in which academics study FLOSS and the reality in which these two groups are (perhaps unknowingly) entangled with one another.

Just as certain subjective notions of FLOSS development, academic methods for the production, inspection and distribution of knowledge are not objective. Academics should therefore take FLOSS development not only as an object of study, but also as an equal in the effort of dealing with the politics of information and knowledge. This goes beyond certain 'open source science' projects and open ways of distributing academic knowledge that have been undertaken by in recent years. Rather, I want to draw attention to their potential fruitful exchange. The most prominent area of potential exchange is most likely the fact that both FLOSS and academic communities rely heavily on the merits of peer review. Academic and FLOSS systems both secure the quality of their products — which is ultimately information — on a peer review basis. Yet, they also deal with its pitfalls. In overcoming these pitfalls they could benefit from interdisciplinary deliberation. It is not my intention to position either one of these two groups above the other. I want to emphasise that their distinct methodologies operate on a same (and subjective) level. Therefore they intertwine and potentially even converge. Rather than seeing them as independent, a lot is to be learnt from the connections between such fields and studies that indicate their exchanges.

▶ Much still needs to be studied when it comes to FLOSS development. Still, I hope this thesis has highlighted potential ways of inquiry. Many of the incumbent academic ideas on, and notions of FLOSS development are very valuable. Yet, now is the time to reinvent them and explore the way we employ them. FLOSS is not another interesting area of study, it teaches valuable and critical insights, musings and even lessons, also to academics. I hope this thesis has made that clear, that it proved to be an interesting read and that it indeed feeds the urge for further and broader inquiry. Thank you for reading.

References

Literature

Appadurai, Arjun. 1990. “Disjuncture and Difference in the Global Cultural Economy.” *Public Cult* 2:1-24.

Bach, Paula M. and John M. Carroll. 2009. “FLOSS UX Design: An Analysis of User Experience Design in Firefox and OpenOffice.org.” In *Open Source Ecosystems: Diverse Communities Interacting*. 237-250.

Bauman, Zygmunt. 1998. *Globalization, The Human Consequences*. New York, NY: Columbia University Press.

Beebe, Barton. 2010. “Intellectual Property Law and the Sumptuary Code.” *Harvard Law Review* 123(4): 809-889.

Benda-Beckman, Franz von. 1995. “Anthropological Approaches to Property Law and Economics.” *European Journal of Law and Economics* 2: 309-336.

Bezroukov, Nikolai. 1999. “Open Source Software as a special type of academic research: critique of vulgar Raymondianism.” *First Monday* 4(10). http://journals.uic.edu/ojs/index.php/fm/article/view/696/606

———. 1999b. “A second look at the cathedral and the bazaar.” *First Monday* 4(12). http://journals.uic.edu/ojs/index.php/fm/article/view/708

Boellstorff, Tom. 2012 “Rethinking Digital Anthropology.” In *Digital Anthropology*. Edited by Heather A. Horst and Daniel Miller, 39-60. London: Berg.

Bollier, David. 2009. *Viral Spiral: How the Commoners Built a Digital Republic of Their Own.* New York: New Press.

Braverman, Harry. 1998. *Labour and Monopoly Capital, The Degradation of Work in the Twentieth Century*. New York, NY: Monthly Review Press.

Bray, Fancesca. 2007. “Gender and Technology.” *Annual Review of Anthropology* 36: 37–53.

Carver, Brian W. 2005. “Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses.” *Berkeley Technology Law Journal* 20(1): 443-481.

Castells, Manuel. 2000. “Toward a sociology of the network society.” *Contemporary Sociology* 29: 693–699.

Chopra, Samir and Scott Dexter. 2007. *Decoding Liberation: The Promise of Free and Open Source Software.* New York and London: Routledge.

Coleman, E. Gabriella. 2001. “High-Tech Guilds in the Era of Global Capital.” *Anthropology of Work Review* 22(1): 28-32.

———. 2004. “The Political Agnosticism of Free and Open Source Software and the Inadvertent Politics of Contrast.” *Anthropological Quarterly* 77(3): 507–19.

———. “The Social Construction of Freedom in Free and Open Source Software: Hackers, Ethics, and the Liberal Tradition.” (PhD diss., University of Chicago 2005).

———. 2010. “The Hacker Conference: A Ritual Condensation and Celebration of a Lifeworld.” *Anthropological Quarterly* 83(1): 47–72.

———. 2013. *Coding Freedom: The Ethics and Aesthetics of Hacking.* NJ: Princeton University Press.

DeWalt, Kathleen Musante, and Billie R. DeWalt. 2011. *Participant Observation: A Guide for Fieldworkers.* Plymouth: Altamira Press.

Ducheneaut, Nicolas. 2005. “Socialization in an Open Source Software Community: A Socio-Technical Analysis.” *Computer Supported Cooperative Work* 14: 323–368.

Eriksen, Thomas Hylland. 2007. *Globalization: The Key Concepts.* London: Bloomsbury Publishing Plc.

Ganti, Tejaswini. 2014. “Neoliberalism.” Annual Review of Anthropology 43 (1): 89-104.

Ghosh, Rishab A. 2005. “Understanding Free Software Developers: Findings from the FLOSS study.” *Perspectives on Free and Open Source Software*. Edited by J. Feller, B. Fitzgerald, S. A. Hissam and K. R. Lakhani. Cambridge: MIT Press.

Grasseni, Cristina. 2004. “Skilled vision. An apprenticeship in breeding aesthetics.” *Social Anthropology* 12(1): 41-55.

Hars, Alexander, Shaosong Ou. 2002. “Working for Free? Motivations for Participating in Open-Source Projects.” *International Journal of electronic Commerce* 6(3): 25-39.

Harvey, David. 2005. *A Brief History of Neoliberalism.* Oxford, NY: Oxford University Press.

Hertel, Guido, Sven Nieder and Stefanie Herrmann. 2003. “Motivation of Software Developers in Open Source Projects: an Internet-based Survey of Contributors to the Linux Kernel” *Research Policy* 32(7): 1159-1177.

Hirsch, Eric. 2010. “Property and Persons: New Forms and Contests in the Era of Neoliberalism.” *Annual Review of Anthropology* 39: 3470-60.

Ho, Karen. 2005. “A view from Wall Street Investment Banks.” *Cultural Anthropology* 20(1): 68-96.

Ingold, Tim. 2001. “Beyond Art and Technology, The Anthropology of Skill” In *Anthropological perspectives on technology* edited by Michael Brian Schiffer. Dragoon, Arizona: Amerind Foundation.

Juris, Jeffrey. 2005. “Violence Performed and Imagined: Militant Action, the Black Bloc and the Mass Media in Genoa.” *Critique of Anthropology* 25(4): 413-432.

———. 2008. Juris, Jeffrey S. *Networking Futures, The Movement Against Corporate Globalization.* Duke University Press.

Juris, Jeffrey and Maple Razsa. 2012. Occupy, Anthropology, and the 2011 Global Uprisings." Hot spots, *Cultural Anthropology* website, July 27, 2012. https://www.culanth.org/fieldsights/63-occupy-anthropology-and-the-2011-global-uprisings.

Kelty, Christopher M. 2004. “Culture’s Open Sources: Software, Copyright, and Cultural Critique.” *Anthropological Quarterly* 77(3): 499-506.

———. 2008. *Two Bits: The Cultural Significance of Free Software.* Durham, NC: Duke University Press.

———. 2011. Inventing Copyleft. In *Making and Unmaking Intellectual Property: Creative Production in Legal and Cultural Perspective*. Edited by Mario Biagioli, Peter Jaszi, and Martha Woodmansee. 133—48. Chicago: University of Chicago Press.

Kottak, Phillip Conrad. 2006. *Cultural Anthropology, Appreciating Cultural Diversity*. New York, NY: McGraw-Hill Education.

Lakhani, Karim R., and Robert G Wolf. 2005. “Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects.” In *Perspectives on Free and Open Source Software*. Edited by J. Feller, B. Fitzgerald, S. Hissam, and K. R. Lakhani. Cambridge, Massachusetts: MIT Press.

Lave, Jean & Etienne Wenger. 2002. “Legitimate peripheral participation in communities of practice” In *Supporting Lifelong Learning* edited by Roger Harrison, Fiona Reeve, Ann Hanson and Julia Clarke. London: Routledge Falmer.

Leach James, Dawn Nafus, Bernhard Krieger. 2009. “Freedom imagined: morality and aesthetics in open source software design.” *Ethnos* 74(1): 51–71.

Lerner, Josh, and Jean Tirole. 2001. “The Open Source Movement: Key Research Questions.” *European Economic Review* 45(4–6): 819–26.

Mansfield, Becky. 2007. “Privatisation: Property and the Remaking of Nature-Society Relations. Introduction to the Special Issue.” *Antipode* 39(3): 393-405.

Mateos-Garcia, Juan, and Edward W. Steinmueller. 2008. “The Institutions of Open Source Software: Examining the Debian Community.” *Information, Economics and Policy*. 20(4): 333-344.

Mazzarella, William. 2004. “Culture, Globalization, Mediation.” *Annual Review of Anthropology* 33: 345-68.

Miller, Daniel & Don Slater. 2000. *The Internet: An Ethnographic Approach.* Oxford: Berg.

Nader, Ralph. 2011. “Going to the Streets to Get Things Done” In *This Changes Everything: Occupy Wall Street and the 99% Movement* edited by Sarah van Gelder. Beret Koehler: San francisco California.

Oreg, Shaul, and Oded Nov. 2007. “Exploring Motivations for Contributing to Open Source Initiatives: The Roles of Contribution Context and Personal Values.” *Computers in Human Behavior* 24 (2008): 2055–2073

Rabinow, Paul. 1997. *Reflections on Fieldwork in Morocco.* Berkeley: University of California Press.

Raymond, Eric. 1999. “The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary.” *First Monday* 3(3). http://ojphi.org/ojs/index.php/fm/article/view/578/499

Robles, Gregorio, Jesus M Gonzalez-Barahona, Martin Michlmayr. 2005. “Evolution of volunteer participation in libre software projects: evidence from Debian.” P*roceedings of the 1st International Conference on Open Source Systems, Genova, 2005.* Edited by Marco Scotto and Giancarlo Succi. 100-107.

Sade-Beck, Liav. 2004. “Internet ethnography: Online and offline.” *International Journal of Qualitative Methods* 3(2) http://www.ualberta.ca/~iiqm/backissues/3_2/pdf.

Schiffer, Michael Brian. 2001. “Toward an Anthropology of Technology” In *Anthropological perspectives on technology* edited by Michael Brian Schiffer. Dragoon, Arizona: Amerind Foundation.

Slukka, Jeffrey. and Antonius Robben. 2007. *Ethnographic Fieldwork: An Anthropological Reader.* Oxford: Blackwell Publishing Ltd.

Söderberg, Johan. 2002. “Copyleft vs. Copyright: A Marxist Critique.” *First Monday* 7(3) http://firstmonday.org/article/view/938/860#s6

———. 2008. *Hacking Capitalism, The Free and Open Source Movement.* New York and London: Routledge.

Stallman, Richard Matthew. 2002. *Free Software, Free Society. Selected Essays of Richard M. Stallman*. Boston, MA: Free Software Foundation.

Sullivan, John L. 2011. “Free, Open Source Software Advocacy as a Social Justice Movement: The Expansion of F/OSS Movement Discourse in the 21st Century.” *Journal of Information Technology & Politics* 8(3): 223-239.

Tsing, Anna Lowenhapt. 2000. “The Global Situation.” *Cultural Anthropology* 15(3): 327-360.

———. 2005. *Friction: An Ethnography of Global Connection.* Princeton: Princeton University Press.

Weber, Steven. 2004. *The Success of Open Source.* Cambridge, MA: Harvard University Press.

Wenger, E. 1998. Communities of Practice: Learning, Meaning, and Identity. Cambridge: Cambridge University Press.

Wilson, Samuel M. and Leighton C. Peterson. 2002. “The Anthropology of Online Communities.” *Annual Review of Anthropology* 31: 449-467.

Zeitlyn, David. 2003. “Gift economies in the development of open source software: Anthropological reflections.” *Research. Policy* 32(7): 1287-1291.

Websites

Apache.
http://www.apache.org/

Free Software Foundation

http://www.fsf.org

Free Software Foundation Europe.

https://www.fsfe.org

Debian.

https://www.debian.org

GNU project.

https://www.gnu.org

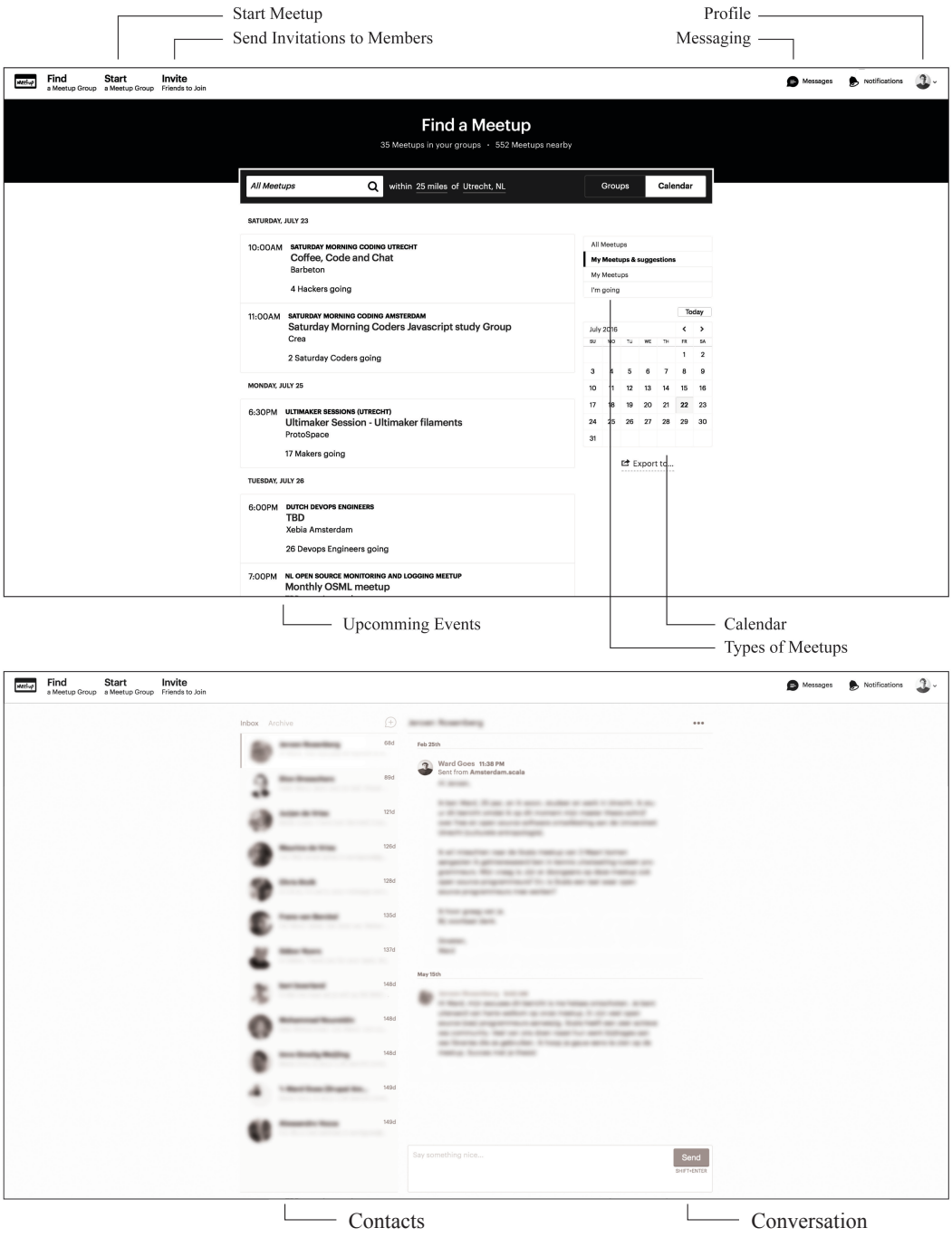
Open Source Initiative.

https://www.opensource.org

Appendices

Appendix I

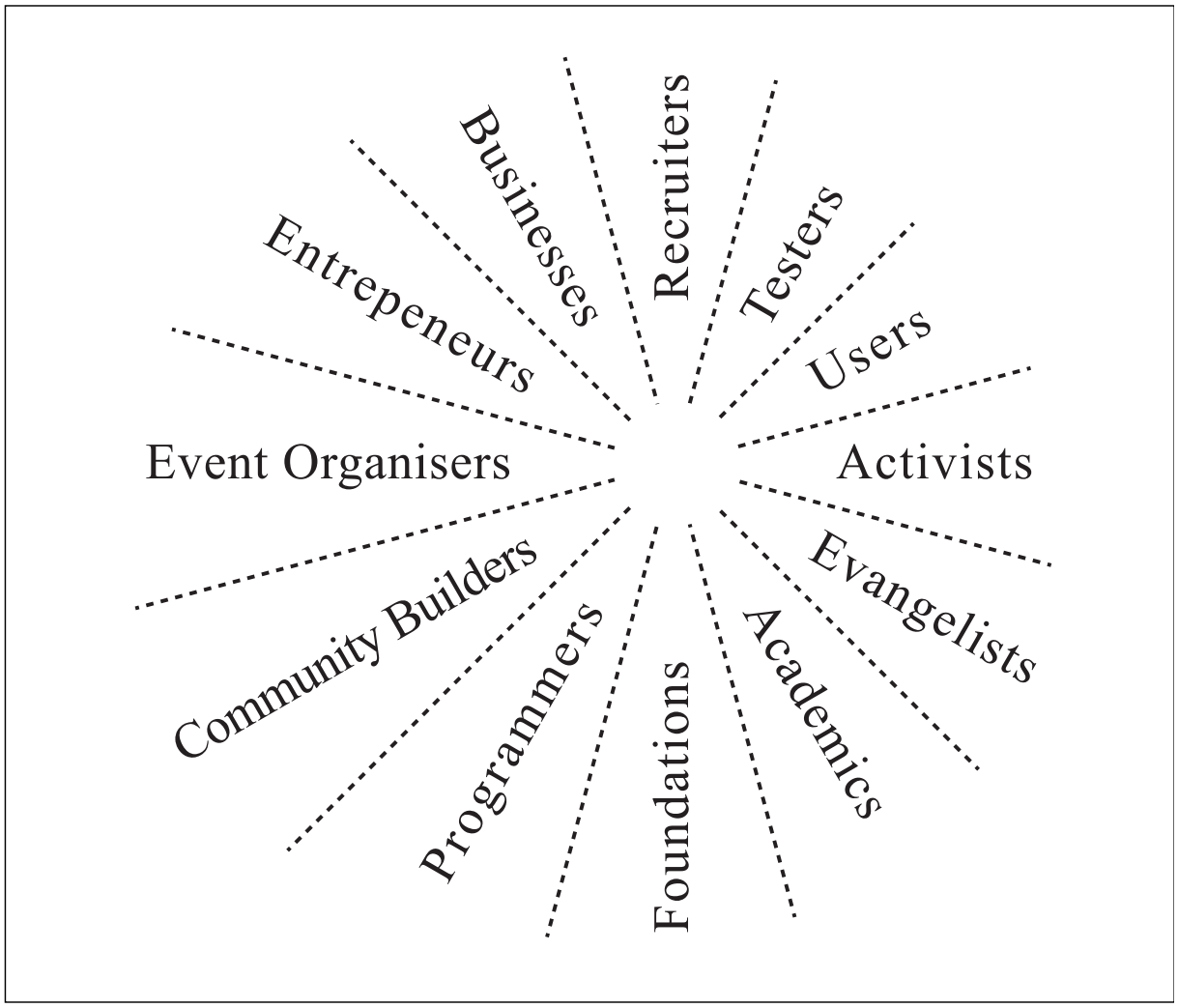
Meetup Interface and Functionality.



Appendix II

Practices and Practitioners in FLOSS development field.

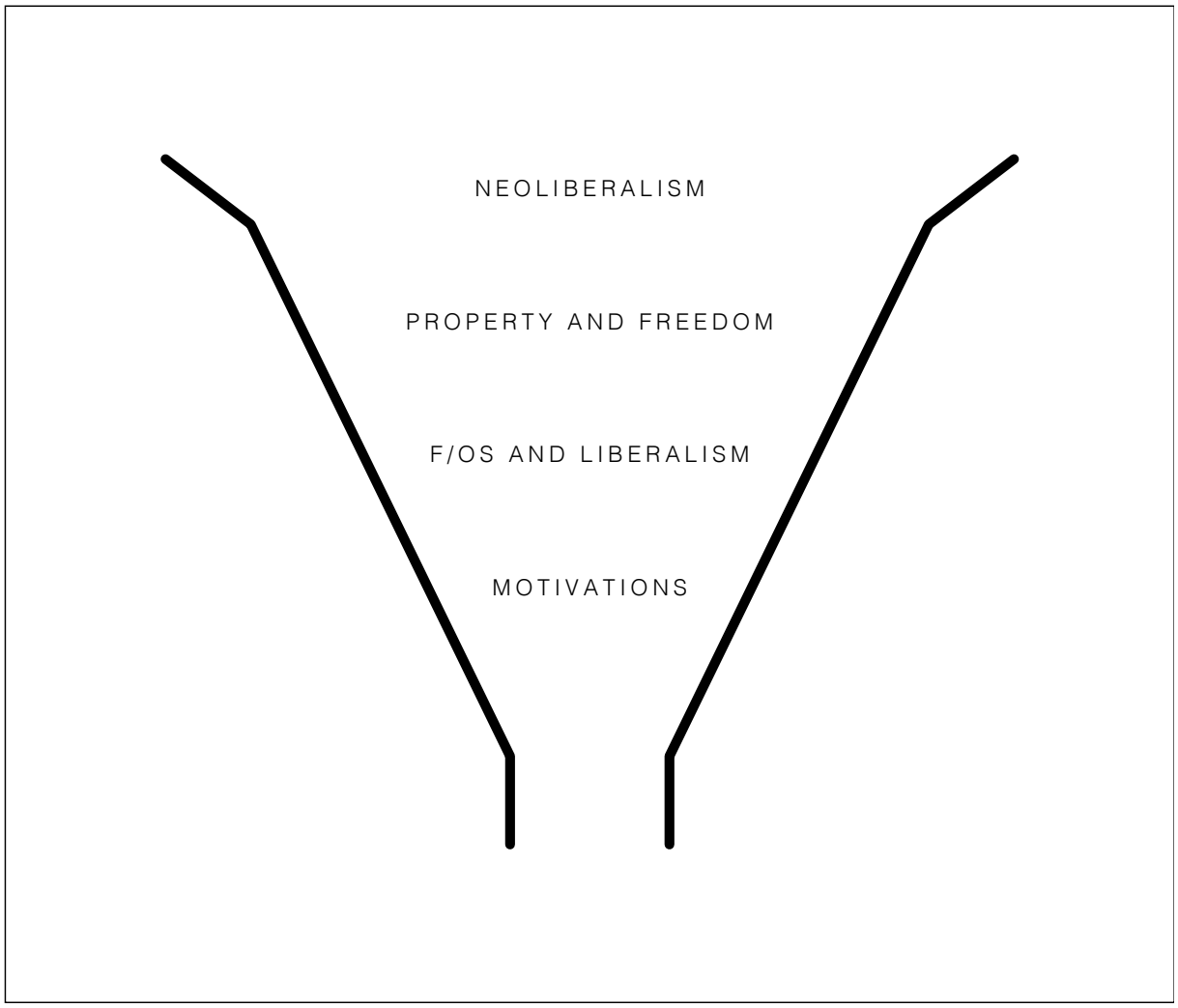
Image: Ward Goes.



Appendix III

Structure of current discourse on FLOSS development in relation to personal motivations of FLOSS developers.

Image: Ward Goes



Read Between the Lines of Code — Perspectives on Free and Open Source Software Development in Perspective is a master thesis for the completion of the master *Cultural Anthropology: Sustainable Citizenship* at the *University of Utrecht*, Faculaty of Social Sciences.

Ward Goes

Abstract —— The principles of free and open source software development are academically understood to be at odds with property- and copyrights; some of the very fundamental principles of contemporary neoliberal society. As a consequence free and open source software development is mostly studied and discussed by academics within this context. This thesis demonstrates that however this is a logical and legitimate approach to studying such software development practices, it is merely one way of conceptually contextualising free and open source software development. This particular contextualisation overlooks the complexity, hybridity and diversity of practitioners, practices and perspectives in the field. Drawing on empirical data, collected during 15 weeks of ethnographic fieldwork, it is argued in this thesis that it is more productive to break with this singular conceptual contextualisation of free and open source software development and explore the variety of contextualisations as found in the field. It reconstructs how together these perspectives renew academic discourse on free and open source software development This will offer a new understanding of how the practices concerning free and open source software development are connected to larger discourses.