

Programmed sociality: A software studies perspective on social networking sites

Taina Bucher

A dissertation submitted to the University of Oslo in accordance with the
requirements of the degree of Doctor of Philosophy in the Faculty of
Humanities

May, 2012

Acknowledgements

This dissertation would not have been possible without the ongoing support of my family and my partner, and the many great, helpful and inspirational people I have had the fortune to meet along the way.

First, I must thank my supervisor Anders Fagerjord and co-supervisor Geert Lovink for their support and helpful criticism. Anders has been an encouraging and positive advisor, whose calmness and pragmatic advice have helped me keep my feet on the ground. I am grateful for the dedicated guidance that Geert has shown throughout the entire project, for his inspiring mentorship, for always pushing my ideas and helping my feet leave the ground again.

This project has greatly benefited from my two research stays in New York City in 2010 and Toronto in 2011. I am extremely grateful for having had the opportunity to experience such intellectually stimulating environments and for meeting such inspiring scholars during these stays. I would like to thank the people who made it possible, for their generous efforts and hospitality.

Thanks to Alex Galloway and the Department of Media, Culture and Communication for inviting me to spend a year at NYU as a visiting scholar. New York turned out to be as great as one could possibly hope for, much thanks to the inspiring seminars, conversations and lectures I had the pleasure of attending while at NYU. I would also like to thank the Faculty of Humanities, University of Oslo, for the financial support that made this formative part of the project possible.

I am thankful for the hospitality shown by Greg Elmer and Ganaele Langlois in inviting me to spend the spring of 2011 at the Infoscape Research Lab at Ryerson University, Toronto. There is a most collegial and collaborative research lab. I would like to thank Ganaele and Greg and the rest of members of the lab, especially Alexandra Renzi and Fenwick McKelvey, for making me feel like part of the team. My work has benefited greatly from the encounter and the discussions I have had with all of them.

A lot of people have read through previous drafts and sections of this dissertation, and very generously provided me with helpful comments and advice. Thanks to: David M.

Berry, Patrick Crogan, Jon Inge Faldalen, Sam Kinsley, Fenwick McKelvey, Georg Kjøll, and the participants of the monthly media aesthetics seminars at the Department of Media and Communication, University of Oslo. As some of the chapters were originally written as articles for different journals and conferences, I would also like to thank the various anonymous reviewers involved for providing useful comments and suggestions that helped to improve the dissertation.

My fellow PhD's also deserve a thank you, especially Jon Inge Faldalen, Henry Mainsah, Cuiming Pang and Gry Rustad who have been there with emotional and intellectual support at the various stages of the process.

Importantly, I have to thank my informants, the many Twitter third-party developers and programmers, for their generosity of time and spirit. Thanks also to Alex Kempton for his help with writing examples of code for the dissertation, as well as taking time to teach me and answer my questions about software and programming.

I would also like to give my sincere thanks to Ann Kunish for her work with the language edit of large portions of this dissertation.

Finally, I would like to thank my family and friends, especially my mother Aira Bucher for her unconditional support and encouragement. Most of all, this dissertation is dedicated to my partner Georg Kjøll. Thank you for always challenging me, making my ideas better, for endlessly reading and discussing my work, for coming to New York and Toronto with me, for listening to me whine about the dissertation always and again (even for hours the day before your own viva), for making me dinner every single day, for always making me laugh, for being the best support and friend anyone could wish for. Thank you Georg, this is for you.

- Oslo, May 2012

List of tables and figures

Table 1 Sample of interview questions	88
Figure 1 Levi's online store. Screen shot from May 9, 2010.	100
Figure 2 EdgeRank formula.....	122
Figure 3 The most recent feed settings with default set to 'Friends and pages you interact with the most'. Screen short from February 25, 2011.	123
Figure 4 Communication stories. Screen shot from September 20, 2011.	129
Figure 5 Facebook welcome. Screen shot April 1, 2011.	144
Figure 6 People you may know. Screen shot from November 18, 2011.	147
Figure 7 'See friendship' page. Screen shot from November 7, 2010.....	153
Figure 8 Sponsored stories. Screen shot from November 19, 2011.....	157
Figure 9 Twitter as of November 27, 2006. Source: Internet Archive	172

Contents

Acknowledgements.....	3
List of tables and figures.....	5
Contents	7
Chapter 1. Introduction	9
The power of social media.....	13
Towards a diagrammatics of software.....	16
Research questions and structure of the dissertation	21
Chapter 2. Theoretical and conceptual context.....	25
Software Studies	26
Medium theory and the materiality of media.....	30
Process-relational philosophy	36
Power: A Foucauldian account.....	42
Chapter 3. Perspectives on software	49
Software: A shifting nexus of relations	51
Code.....	52
Algorithm.....	58
Executable and execution	62
Practice.....	64
Chapter 4. Methodological framework.....	69
Technography: A descriptive-interpretative approach.....	70
Letting the software ‘speak’.....	74
Interviewing developers.....	82
Chapter 5. Open Graph protocol: Arranging attention	93
Paying attention to attention	95
Arranging attention: The case of the Facebook platform	97
A technicity of attention.....	105
Chapter 6. EdgeRank algorithm: Becoming (in) visible.....	113
Media visibility	116
Algorithmic visibility.....	120
Rethinking regimes of visibility.....	124
Participatory subjectivity	132
Chapter 7. Managed relationships: Assembling friendship.....	137

Revisiting friendship online	140
Software-generated friendship	143
Towards an understanding of algorithmic friendship	150
Chapter 8. The Twitter APIs: Objects of intense feeling.....	163
Application Programming Interfaces	167
The Twitter APIs.....	171
Community of practice	175
The construction of collectives: APIs as quasi-objects.....	182
Governing innovation	186
Chapter 9. Conclusion: Turning the tables on ‘digital humanities’	193
Major contributions.....	194
Thinking ahead: Social media and software studies	200
References.....	205
Appendix 1. List of interviews.....	221

Chapter 1. Introduction

This dissertation is about the increased power of code and algorithms in the social fabric of everyday life. It explores how the software processes and mechanisms of social networking sites establish certain forms of sociality, specific to how they produce the conditions for the sensible and intelligible.

It is not even six years ago that I became a member of Facebook, in October 2006, when the social networking site opened to all Internet users. At that time Facebook had about 8 million users, today it has 900 million – the numerical equivalent of the entire Internet population back in 2005. What started as a social networking site for a closed circle of Ivy League and Stanford students has turned into one of the most popular places for people on a global scale to gather and spend time online. It is the place to check for updates on what one's friends have been up to, form new attachments, share things, engage in conversations, and get new information.

Ever since Facebook's opening up to the general Internet population, research on social media has followed suit and seen a comparable surge to that of Facebook's user mass. The result of this is that we now know a great deal about the nature of social life on Facebook and other social media platforms, for instance how users interact and form identities online (i.e. Baym, 2010; boyd, 2007). Much less is known, however, about the material and infrastructural support of social media. How does the software underlying social media platforms establish the condition of possibility for sociality online? What power relations exist between software architectures and users? Although there is more work emerging in this era (i.e. Langlois et al., 2009b), as it stands, we simply do not know enough about the articulation of power in and through software embedded contexts of social media.

As José van Dijck observes, platforms like Facebook are not just tools for facilitating connections; they are also makers of 'algorithmically configured connections - relationships wrapped in code - generating a kind of engineered sociality' (2012: 161-162).

My project starts out from an observation and recognition of the fact that software increasingly plays an important role mediating and governing everyday life. Following from this, the main objective of the dissertation is to develop an

understanding of the ways in which software not just makes sociality possible, but how software needs to be understood as a key actor in shaping specific ways of relating to self and others in the context of social networking sites. The intent is to contribute to a widening of the disciplinary focus of media and communications, by investigating social networking sites from a software-sensitive perspective. Specifically, this project is meant to demonstrate how software can be studied as a key site for an understanding of sociality online.

The chief contributions of this dissertation are: 1) to offer an understanding of the micropolitics of power in a software dense mediascape; 2) to show how the concrete ways in which software – specific algorithms, protocols, and features – can be analysed as producing the conditions for the sensible and intelligible, relying on a reading of software mechanisms and empirical online interviews with key actors in the social media industry; and 3) to provide an account of programmed sociality - how sociality in social networking sites is algorithmically and dynamically shaped around the pursuit of participation.

The project begins from two interrelated assumptions. The first is that there is a lack of scholarly attention paid to the materiality and medium-specific features of software that influence sociality online within the field of media and communications. This first assumption is based on work I did as a research assistant at the Department of Media and Communication, University of Oslo, following my master's thesis on discourses of participatory culture at the London School of Economics and Political Science (LSE). In Oslo, I spent several months on an extensive literature review of the rather broad field of new media use. What I found was an enormous amount of existing literature on motivation and usage practices pertaining to all kinds of online and mobile media, with an overwhelming focus on understanding the engagement with such media from a user perspective. Little work had, however, been done on the medium itself. Some of the questions that did not seem to find any answers in the literature had to do with what the structural aspects and the technical mechanisms and operational logics underlying social media use were.

The second assumption is that software matters and an understanding of which should form a part of any analysis of how sociality is established online. This assumption evolved out of my realisation described above, and coincided with some of the first organised attempts to establish the field of *software studies* during the spring of 2008.

The efforts to establish software studies as a field of study, drawing on literature and theories from computer science, sociology, cultural studies, media studies, and literary studies, immediately resonated with some of the issues I had been considering. So far a main focus of software studies has been on the importance of delineating and conceptualising the field itself. While the scope and potential of its application is still not fully felt, much theoretical, methodological, and empirical work remains to be undertaken.

The aim of this dissertation is thus two-fold. First, it seeks to introduce a software-sensitive approach to new media studies. By investigating how power articulates through software in configuring sociality online, it makes a contribution to the field of media studies and particularly to scholarship on social media. Here I draw on a Foucauldian understanding of power as productive, which I will discuss in more detail in the next chapter. My interest in the ‘productiveness’ of software naturally feeds into a more theoretical and analytical interest in how scholars within the humanities and social scientists might begin to make sense of software as an object of study. Inspired by recent endeavours in software studies and related disciplines, this project also seeks to contribute to these efforts by offering both a theoretical and empirical investigation of a particular type of software, namely social networking sites, such as Facebook and Twitter. To my knowledge, this study is one of the first explicitly concerned with social networking sites from a software-studies perspective. As such, this project will necessarily have to undertake some ground clearing of conceptual and methodological issues as part of an examination of what is essentially seen as a study of *programmed sociality*.

In regard to the term ‘programmed’, I wish to draw attention to the way in which software prescribes certain norms, values, and practices. In addition, I will use computer scientist John von Neumann’s notion of program, where the term ‘to program’ means to ‘assemble’ and to ‘organise’ (see Grier, 1996: 52). In terms of ‘sociality’, I refer to the concept of how different actors belong together and relate to each other. That is to the ways in which groups of entities (both human and non-human) are gathered into specific forms of collective association, enabling interaction between the entities concerned (Latour, 2005). Thus, to be concerned with programmed sociality entails an interest in how actors are articulated in and through

coded means of assembling and organising, which always already embody certain norms and values about the social world.

By drawing these concepts together, I wish to develop a critical understanding of the ways in which sociality and subjectivity are shaped in and through the material-discursive conditions of software in the context of social networking sites.¹ Importantly, a central premise here is that sociality is not merely confined to human relationships in the classic sociological sense. For example, Karin Knorr Cetina (1997) argues persuasively for an expanded understanding of sociality by including material objects in what she terms an ‘object-centred sociality’. In this sense, relations to self and others online are increasingly mediated, processed, and even engineered by the complex dynamics of software that make up the specificities of the computational environment. Thus, this dissertation seeks to contribute an understanding of the types of forces and mechanisms at play in the articulation of sociality online.

Broadly stated, I am concerned with the question of how software signifies and is suggestive of things, and how software ‘makes sense’ in networked environments. On the one hand, this question refers to the meaning of software and how software makes sociality meaningful in specific ways. On the other hand, it asks how one might make sense of software itself, and how software in turn produced the conditions for the sensible and intelligible. These are seen as fundamentally intertwined questions that both need to be considered in order to be able to see what software is, what it does, and which relations it helps to forge in the context of social networking sites. As Adrian Mackenzie suggests, ‘perception of materiality and the materialities of perception figure centrally in relation to software’ (2006: 173). Thus the question of how software signifies and makes sense has a double logic. The first refers to the ontological and epistemological dimensions of software itself, which ask for its nature and how one can make sense of it. While this dissertation will touch upon the ontological and epistemological dimensions raised by studying software, the focus is on how software produces new forms of sociality. Social networking sites are not empty spaces upon which sociality and subjectivation simply occur. Software contains certain normative and prescriptive structures. What we see and what we can

¹ For an explanation of the material-discursive conditions with regards to media technologies and software, see Chapter 2.

know by using and being on sites such as Facebook and Twitter, is the result of complex material-discursive practices involving both human and non-human actors.

This project is inscribed within the larger field of media studies; not simply because of the nature of its empirical objects of study – social networking sites – but also because software is seen as a medium, that is, as a material basis for mediation and expression.² In other words, seeing software as a medium in this sense means that software is capable of doing things, of expression and communication. Not only does software act as a vehicle of information, it embodies ways of encoding information that are articulated differently in text, image, and sound. A core tenet of media studies is the notion that media are never neutral carriers of meaning; they transform, translate, persuade, shape, construct, and produce (see for instance Mitchell and Hansen, 2010). This leads to another important way in which my study on programmed sociality is firmly grounded within media studies: namely, through its concern with power.

The power of social media

This dissertation is concerned with the specific case of social networking sites - bounded web-based software systems that allow users to connect socially and form networks. While there is no one precise definition of social networking sites (SNS), boyd and Ellison (2007) have identified three key characteristics: the construction of a public or semi-public profile within a bounded system; the articulation of a list of other users with whom they share a connection; and the possibility to view and

² Media studies offers a rich vocabulary for theorising and understanding software in terms of its communicative, performative, and encoding capacities. While I will not embark on an explication of these terms here, it is worth noting how the notion of medium offers a useful way to view software critically. Rather than seeing software simply as the product of mechanical (en)coding reminiscent of mainstream computer science (there are of course notable exceptions to the seeming lack of criticality within computer science; see for instance Donald Knuth's 'The Art of Computer Programming' (1968)), a media-studies perspective makes it possible to critique software on a variety of different dimensions, including but not limited to: communicative (as in who shares what with whom, of what is said or expressed in what ways), cultural (as in processes of shaping meaning, as sites of public and personal struggles, or a set of shared collective values), aesthetic (how it provides the condition for modes of appearance and perception), and technical (as in potential for storage, transmission, and processing). See for instance Hansen and Mitchell's *Critical Terms for Media Studies* (2010) and Guillory's 'Genesis of the Media Concept' (2010) for good conceptual accounts of the medium.

traverse their list of connections and those made by others within the system. Social networking sites hit the mainstream around 2003 with sites such as Friendster and MySpace. Since then, a plethora of such sites has seen the light of the day, but only a few still persist. My dissertation is limited to Facebook and Twitter, currently the two most popular social networking sites.

Facebook and Twitter form part of the wider phenomenon of ‘social media’ and ‘Web 2.0’. Perhaps more so than is the case with the term social networking sites, the definition of ‘social media’ is still very much under debate. Usually, however, social media is used as an umbrella term for a variety of web-based software services that include blogs, wikis, social networking sites, and media sharing sites that leverage on and makes possible the production of user-generated content. The concept of Web 2.0 on the other hand refers to an idea about the Web articulated by the publisher Tim O’Reilly back in 2004 and 2005. The notion of Web 2.0 describes a set of principles and practices defining the Web as a participatory platform, contrasting it to the notion of Web 1.0, seen as a mere information source (see O’Reilly, 2005; Song, 2010). Rather than naming an explicit stack of technologies associated with this new era, O’Reilly proposed Web 2.0 as a new attitude toward, or way of using/leveraging the Web, most notably in terms of software development and end-user use.

Social media are not merely powerful in terms of their user base. The popularity and influence of social media attests to their ability to connect people across time and space. More and more people spend a considerable amount of their time on these sites. According to a recent report by the Pew Internet, ‘Americans spend more time on SNS than doing any other single online activity’ (Hampton et al., 2011: 8). The scholarly debate on the emergence and nature of social media or Web 2.0 can roughly be grouped in terms of belonging to either a positive or a negative camp. The positive camp has tended to herald social media as sites of empowerment, by focusing on what Henry Jenkins (2006) calls ‘participatory culture’, or similarly highlighting the ways in which users have increasingly become ‘producers’ (Bruns, 2008). The more pessimistically-inclined scholars have tended to focus on social media as reinforcing hegemonic power, by pointing to the ways in which user-generated content and the vast amount of participation is ‘exploited’ by neoliberal capitalism, where the concept of ‘immaterial labour’ (Terranova, 2000; Lazzarato, 2004) plays a central role. In many ways, these debates revisit classic media studies debates on the empowering or

disempowering effect of media on users and audiences. Despite the different analytical frameworks employed to understand the impact of social media, a common denominator in research on social media is the concept of *participation*.

According to O'Reilly (2005), participation is intrinsic to the very architecture of social media. The ways in which social networking sites enable connections to be forged, maintained, and performed do not occur on neutral ground. As José van Dijck argues, 'the novelty of social media platforms is not that they allow for making connections but lead to engineering connections' (2012: 168). Not only do social networking sites engineer connections, they arguably provide the very material-discursive conditions for participation itself. This points to the technical side of social media, to the power of code. Lawrence Lessig (2006) infamously claimed that 'code is law' because it regulates the structures through which things can emerge. Code influences the ways in which people can move about and navigate on the Web, affecting what can be said and done online. The regulatory power assumed to be at work through code highlights the ways in which social media can be understood as an architectural structural force.

Having briefly established the hypothesis that social media are powerful, not just in terms of their widespread use, but also in terms of their political-economic context as well as their material constitution in code and software, a principal research question can be formulated. While it is taken for granted that social networking sites constitute and are constituted by power relations, what needs to be addressed is therefore not *if* power articulates in these media spaces, but rather *how*, and with what possible implications. The overall aim of this study, then, is to examine the ways in which *power articulates through software in the context of social networking sites*.³

³ As I am well aware of the notion of 'articulation theory' within cultural studies (as formulated by Stuart Hall, Ernesto Laclau, Chantal Mouffe and others), it is perhaps necessary here to stress that my use of the word 'to articulate' is not meant to signify an adoption or use of these theories within the context of this dissertation. That said, my choice of the word 'to articulate' indeed seeks to signify some of the 'nice double meaning' of the word, as Stuart Hall puts it in an interview with Lawrence Grossberg (Grossberg, 1986: 53). In this sense, articulation is used as a term that captures both the meaning of expression and connection. While this dissertation does not explicitly rely on articulation theory, I rely on its key insight in terms of being attentive of the ways in which power articulates through software necessitates an awareness of the fact that the relations forged and elements linked through software are not 'necessary, determined, absolute and essential for all time' (Hall in Grossberg, 1986)

More specifically: Through which techniques and mechanisms does software participate in the shaping of sociality online? What kinds of cultural assumptions are embedded in software, and what do they bring to bear? If we follow Michel Foucault's conceptualisation of power as productive,⁴ we may begin to ask what it is that software produces, what software is capable of, how it activates relational impulses; in short, how software operates to shape and govern the conduct of subjects in the participatory culture of social networking sites.

Towards a diagrammatics of software

This research is organised around four separate but interconnected case studies, each of which aims for a critical investigation of the nature of programmed sociality by focusing on exploring the power of software as it operates through the algorithms and protocols of Facebook and Twitter. My interest in questions of power stems from the notion that power is increasingly materialised in and through software, as software and code have become ubiquitous, slipping into the 'technological unconscious' of everyday life (see Lash, 2007; Thrift, 2005). I hold that software has productive capacities, not merely by mediating the world, but through its delegated capacities to do work in the world and by making a difference to how social formations and relations are formed and informed. Situating this dissertation within the still-emerging field of software studies offers an important and much-needed perspective on social media that seeks to challenge the hegemony of usage studies, by questioning and focusing instead on the ways in which software, through its capacities to assemble and organise can be said to allow for, encourage, or block certain kinds of actions (Fuller, 2008: 7). Setting out to explore power through software seems indeed like a daunting and perhaps even impossible task. However, it is not my intention to investigate power as a totalising force that can be located and described once and for all. Rather, as a diagrammatics, or cartography of power, my case studies are meant as explorations of the ways in which the power of software operates on many different levels.

⁴ I will provide an overview on Foucault's notion of power in the next chapter. Suffice it here to say that I do not view power as something repressive, but rather as a relational force capable of affecting and being affected.

I concern myself primarily with two dimensions of software, algorithms and protocols. Algorithms are the coded instructions that a machine needs to follow in order to perform a given task. Protocol refers to a set of conventions governing the transmission and exchange of messages in distributed networks. Both algorithms and protocols can be understood as plans of action or rules that govern computational processes. From a media and communications perspective especially, algorithms and protocols are important elements when considering networked and software-enabled media such as social networking sites, as they in many respects prescribe and define the possible actions within these programmed spaces.

Algorithms not only epitomise the operationality of software; as Mackenzie (2006: 43) suggests, they also participate in defining the orderings of the social field. Algorithms are at the center of our information ecosystem, where they are used to sort, filter, suggest, recommend, summarise, map, and list information and content of the Web according to predefined parameters. Increasingly, we have come to rely on these programmable decision-makers to manage, curate, and organise the massive amount of information and data available on the Web, and to do so in a meaningful way.

As we delegate an ever-increasing amount of tasks to algorithms functioning as automated decision-makers, it becomes imperative to better understand their operational logics. Some of the critical questions that arise from this and which will be further explored in the chapters to come thus include: what role do algorithms play in Facebook? What kinds of cultural assumptions are in fact encoded? How do algorithms configure their users? What forms of sociality do algorithms aspire to emulate? Which associations are made, and what relationalities do algorithms articulate?

Protocol, as Alexander Galloway (2004) has argued, is not merely a technical specification regulating how data can be exchanged on a network. Above all, protocol can be seen as a management style, a technique for managing contingent relations, affecting not only the technical level, but also the social level.⁵ Perhaps most

⁵ The Web involves many different layers of protocol, from the very infrastructural TCP/IP protocol that regulates the transmission of information on the Internet, HTTP that regulates the transmission on the Web, the graphical arrangement of objects in browsers through HTML, to the standardised text encoding protocol of XML (or JSON etc.).

importantly in the context of this study, protocol signifies power, a managerial and governmental form of power. Like algorithms, protocols have the capacity to order the social, to make the world hang together in certain ways and not others. They accentuate relations; make certain things hold together in specific ways.

This raises some interesting questions that will be further explored, including: what is it that protocols hold together, what are the principles of the connections? If we consider protocol as Galloway does, as a management style, then what and who is being managed by whom, and to what effect? And how can we understand the coordinative work that protocols do?

The case studies are organised around specific algorithmic and protocological elements that are seen to play a particularly important role in terms of shaping relations to self and others on Facebook and Twitter. Specifically, I am concerned with the algorithms employed by Facebook that govern participation on the platform, including EdgeRank, GraphRank, and the ‘People You May Know’ algorithm (see chapters 5, 6, and 7). In terms of protocol, understood in the general sense of connecting code, I will be looking specifically at Facebook’s Open Graph protocol and Twitter’s application programming interfaces (see chapters 5 and 8, respectively).

Methodologically, I rely on what I call a *technographic* approach that borrows from hermeneutics, material and discursive methods including close reading, medium-specific analysis, textual analysis, and qualitative interviews. The project follows an associative strategy inspired by Actor-Network Theory (Latour, 2005) and multi-sited ethnography (Marcus, 1995). This implies tracing the associations provided by the objects of study, in order to be aware of the connections that they forge and of what the different actors are capable.⁶ Following Bruno Latour’s notion that ‘things might authorize, allow, afford, encourage, permit, suggest, influence, block, render possible, forbid, and so on’ (2005: 72), the specific articulations discussed as part of the case studies are thus grounded in the software elements themselves, what they suggest, allow for, and afford. Thus, each of the chapters seeks to provide insight into the ways in which power articulates through software, producing new norms of sociality and

⁶ An actor is here understood in the sense of ANT as any element that makes a difference to some other elements course of action. I will explicate this in more detail in the next chapter. What is important here is the fact that actors are not limited to human actors, but also refers to non-humans.

connectivity. As a whole, this dissertation can be read as a *diagrammatics* of the programmed sociality of social networking sites.

Derived from Gilles Deleuze's reading of Foucault, the concept of *diagram* signifies a map of power. Deleuze (2006) suggests that Foucault should be understood as a new cartographer, someone intent on mapping out the relations between forces, to show how power produces new realities. Power in this sense is never power-over, but rather power-through and power-between. Power is not possessed but practiced, passing 'through the hands of the mastered no less than through the hands of the masters' (Deleuze, 2006: 60). Diagram is not a static structure of power but rather the function or operationality of power. Following Deleuze's assertion that 'every society has its diagram(s)' (2006: 31), what is at stake here is thus a diagrammatics of software, understood as the cartography of strategies of power. David Rodowick elaborates on the nature of such mapping:

What is rendered visible and thus knowable in an epoch derives from a historical dispositif in every sense of the word: architectural (the Panopticon as a disciplinary plan); technological in the sense of a strategic arrangement of practices or techniques; but also philosophical or conceptual (1990: 18).

Foucault's analysis of the architectural, technological, and conceptual nature of diagrams provides a useful methodological and analytical framework for this study. While only one of the case studies (Chapter 7) makes explicit use of the architectural framework offered by Foucault's writings on the Panopticon, the dissertation overall draws upon the ideas and analytical cartography reminiscent of Foucault's writings in general. An architectural perspective usefully highlights the ways in which spaces are 'designed to make things seeable, and seeable in a specific' way (Rajchman, 1988). This, I argue, offers a useful avenue to analyse software in terms of architectural structuring, where embedded technical means have the power 'to incite, to induce, to seduce, to make easy or difficult, to enlarge or limit, to make more or less probable' (Foucault quoted in Deleuze, 2006: 59).

The notion that code and software have a regulating power can perhaps better be understood if we consider it in relation to Foucault's concept of *government* and *governmentality*. While Foucault used these terms somewhat interchangeably, especially in his later writings, government broadly refers to the 'conduct of conduct', whereas governmentality can be understood as the modes of thought, or rationalities,

underlying the conduct of conduct (see Foucault, 1982; 2007; 2008; Lemke, 2001; Rose, 1999).⁷ For a mapping of power, the notion of government offers an elaborate concept for the kinds of architectural shaping that Foucault described in *Discipline and Punish* (1977), as it points to the multiple processes, measurements, calculations, and techniques at play in organising and arranging sociality. Governmentality has a crucial technological dimension, as the conduct of conduct can be achieved through various technical means.⁸ As such, governmentality becomes a useful concept that highlights how software ‘has become a technology of government’, as it ‘consists of rules of conduct able to be applied to determinate situations’ (Thrift, 2005: 172). Analytically, this implies a focus on the ways in which software arranges and organises, its specific techniques and procedures, and the mechanisms used in processes of individual and collective individuation.⁹

Through what I call a *software sensitive analytics*, the architectural and operational logics of social networking sites are read not so much for their hidden meaning as a traditional hermeneutics would have it, but for their ‘medium specificity’ (Hayles, 2004). In terms of software, such a mode of critical attention implies an analysis that takes into account the specificities of algorithms, protocols, code, and features. Software ‘makes sense’ by producing regimes of visibility (a key category of governmentality), by making something intelligible and thus knowable. Following Matthew Fuller, I will attempt to demonstrate some of the ways in which software exists and can be experienced, ‘to show, by the interplay of concrete examples and multiple kinds of accounts, the condition of possibility that software establishes’ for sociality online (2008: 1).

⁷ For a good discussion on the concept of government and governmentality, see Michel Senellart’s overview in Foucault, 2007: 499-507.

⁸ As Bröckling et al. point out, such technical means may for instance include social engineering strategies embedded in various machines, medial networks, recording and visualization systems (2011: 12).

⁹ The dimension of subjectivity was key for Foucault, who used the concept of governmentality to analyse everything from the production of orderly and compliant ‘docile bodies’ through pastoral guidance techniques (see Foucault, 1978) and the emergence of liberalism, in which notions of freedom are produced so as to replace external regulation by inner production (Bröckling et al., 2011: 5).

Research questions and structure of the dissertation

To recapitulate, this study has two overarching research questions that provide the overall focus. First, I am interested in the ways in which software signifies and is suggestive of things, and how it ‘makes sense’ in networked environments, in terms of producing conditions for the intelligible and sensible. In many respects this is a question of power relations. Thus, my second question pertains to *how* and *with what implications* power articulates through software in the context of social networking sites. The core of the dissertation is organised around four case studies that address issues I see as particularly important in the present context of the programmed sociality of social networking sites. These investigations look at the ways in which software codifies modes of attention and constructs regimes of visibility, and how software manages friendships and catalyses specific developer practices and discourse. While all of the chapters address the overarching research questions, each chapter addresses a different body of theoretical concerns and questions.

Chapter 2 situates the project within the fields of software and media studies. It provides a literature review and overview of previous research, and analytical perspectives relevant to the thesis. First, it introduces the readers to the field of software studies by focusing on existing literature that explicitly seeks to understand social media from a software studies perspective. Second, I situate the thesis within the branch of media studies that most notably has been concerned with the specificity and materiality of the medium, focusing on the influence of Marshall McLuhan and Friedrich Kittler. The chapter furthermore provides an overview of some of the theoretical perspectives and concepts used to frame the case studies. The thesis builds on what can be termed a process-relational theoretical framework, including actor-network theory and assemblage theory. Finally, the notion of power is introduced, derived from the work of Michel Foucault.

Chapter 3 provides a conceptualisation of software by discussing the nature of software as code, algorithms, execution and practice. In order to comprehend the ways in which software signifies and is suggestive of things, and how it ‘makes sense’ in networked environments, I argue for a conceptual understanding of the nature of software. The argument is frequently made that media studies needs to be more attentive to matters concerning software. This chapter therefore offers some perspectives on what I take to be necessary (but not sufficient) conditions of software

that are important to an understanding of programmed sociality. In doing so, this chapter expands on the literature and field of software studies briefly introduced in Chapter 2, to provide a more detailed account of some of the prevailing debates and discourses around software. This chapter thus seeks to address the overarching research question of how software signifies and suggests certain things by grounding such an understanding in the materiality, specificity, and affordances of software.

Chapter 4 outlines the research methodology. The chapter introduces the notion of *technography*, a way of reading technical objects and ensembles by using hermeneutic and ethnographic methods. Technography is understood as a descriptive account of software and its practices, grounded in data attained from multiple sources. In this thesis, it includes data from auto-ethnographic observations of the social networking sites, experiments in ‘reverse engineering’, document analysis of technical specifications, manuals, articles, and engineering talks, as well as online structured interviews with third-party software developers.

Chapter 5 explores the power of the Facebook’s Open Graph protocol in terms of governing attention. This chapter thus addresses the question of how software ‘makes sense’ in terms of its arranging and organising capacities. The discussion is grounded in a close reading and description of Open Graph. The question is how software is used to govern attention on and through the Facebook platform? Tracing the infrastructural developments of the Facebook platform, I argue that there is a shift from an object-oriented attention economy reminiscent of the ‘like button’, to a form of anticipatory and personal attention economy produced by the performativity of algorithms and software-enabled personalisation processes. Not only is the operational logic of the software anticipatory, it invokes a form of anticipation that seeks to realise its future. Using the concept of *technicity* as an analytical framework, the argument is made that attention needs to be seen as emergent, enabled by the unfolding power of software to produce the conditions for the sensible, in conjunction to users.

Chapter 6 explores the notion of *regime of visibility* on Facebook, which I argue is algorithmically constructed. It takes Facebook’s EdgeRank algorithm as a case in point to analyse the ways in which software shapes a characteristic form of visibility materialised in the News Feed. The question addressed is how we can begin to understand this kind of algorithmic intervention, through which specific politics of

arrangement, architecture, and designs. In this chapter, I suggest that the logic of the News Feed works in reverse to Foucault's notion of surveillance, as developed in *Discipline and Punish*. Examining EdgeRank's operational logic by means of an experiment of 'reverse engineering', the argument is made that participatory subjectivities are produced not through the instalment of visibility as a constant and threatening possibility, but rather through what I call the 'threat of invisibility', whereby visibility functions as a reward as opposed to Foucault's notion of punishment.

Chapter 7 investigates the formation of friendship online. The question raised is how the software acts to encourage and support a particular kind friendship on Facebook, and which possibilities the software offers for friendship performance. In line with Foucault's notion that power is first and foremost productive of subjects, this chapter investigates the sociotechnical shaping of friendship, seen as one of the most prevailing forms of sociality encouraged by social networking sites such as Facebook today. In this chapter, I argue that friendship on Facebook exists as a relation between *multiple actors*, not only human individuals, and importantly involves algorithmic and commercial relations of forces.

Chapter 8 examines the power relations permeating *Application Programming Interfaces* (APIs). Framed as protocological software objects, the question here is how the APIs form and hold relations together and how can we understand the coordinative work that protocols do? Using the Twitter APIs and its third-party developer ecosystem as my case study in the chapter, I argue that APIs constitute objects of 'intense feeling', which allow not only the flow of information and data, but also articulate various relations of control and freedom. This chapter presents the empirical findings of the interview-based research on the Twitter API developer community, where multiple valences of software as practice and experience are explored and discussed.

Finally, in Chapter 9, I draw the elements of the dissertation together in order to outline how we can further develop the notions of sociality and software that I have presented, and provide suggestions for further work.

Chapter 2. Theoretical and conceptual context

In this chapter, my aim is to outline the literature and theories relevant to the question raised by software within the context of understanding and studying social media. Here I concentrate on providing a comprehensive account of the different theoretical trajectories and fields that are particularly important to this study. This includes an outline of the growing field of ‘software studies’; an account of more established ways of viewing media studies in terms of ‘medium theory’, and related concepts such as the materiality of media and communication; and an account of what can be termed ‘process-relational philosophy’. Finally, this chapter provides an overview of the concept of power as it is used throughout the thesis, based on the writings of Michel Foucault.

Considered together, medium theory, the materiality of media, process-relational philosophy, and Foucault’s conception of power provide, what I believe, is a fertile theoretical and conceptual framework for studying software and its practices. As a nascent field of study, the definition and contours of software studies are still very much up for grabs. In my view, software needs to be understood as a material-discursive phenomenon, meaning that software produces conditions for the intelligible and sensible that are always socio-historically grounded in specific material supports, or rather, material (re)configurations. I take a non-representational stance in regard to software in the sense described by Nigel Thrift, in that the non-representational is an approach to understanding the world in terms of *effectivity* rather than representation: not the what, but the *how* (2008: 113, my emphasis). It implies beginning from the assumption that media technologies produce specific regimes of power, and that one useful way to begin to analyse this power goes via the workings of specific technologies, in order to uncover the ‘systems of rules that govern its functioning in the first place’ (Gane, 2006: 78). Rather than debunking existing approaches, I will use this chapter to present what I take to be a constructive and productive method for the conceptualisation of software studies, which exists at the intersection of materialist approaches to media, assemblage theory, nonhuman agency, and power relations.

Software Studies

While still in its infancy as an academic field, software studies constitutes a particularly important field for situating this study, as it makes an explicit point about the need to pay attention to the many different scales of software in the analysis of digital culture and network societies. The concept of software studies first appeared as part of Lev Manovich's argument concerning the programmability of new media in his book *The Language of New Media* (2001). As Manovich put it:

New media calls for a new stage in media theory whose beginnings can be traced back to the revolutionary works of Harold Innis in the 1950s and Marshall McLuhan in the 1960s. To understand the logic of new media, we need to turn to computer science. It is there that we may expect to find the new terms, categories, and operations that characterize media that became programmable. From media studies, we move to something that can be called "software studies" - from media theory to software theory (2001: 48).

In a more recent account, Manovich points out that the above quotation from 2001 – which has become a common way of introducing the notion of software studies (at least from a historical perspective) – is in need of some adjustment. As Manovich writes in the introduction of his forthcoming book *Software Takes Command*:¹⁰

Reading this statement today, I feel some adjustments are in order. It positions computer science as a kind of absolute truth, a given which can explain to us how culture works in software society. But computer science is itself part of culture. Therefore, I think that Software Studies has to investigate both the role of software in forming contemporary culture, and cultural, social, and economic forces that are shaping development of software itself (2011: 6).

This quotation more accurately describes what I believe the notion of software studies encompasses: namely, the need to shed light on an object long overlooked within media and cultural studies. In this sense, the emergence of software studies has been framed in terms of identifying a 'new object of study and area of practice for kinds of thinking and areas of work that have not historically "owned" software' (Fuller, 2008: 2). In Manovich's terms this means:

¹⁰ A first version of this book was published November 20, 2008 online, under a creative commons licence, and has been available for download here: <http://lab.softwarestudies.com/2008/11/softbook.html>. Here I am quoting a July, 2011 version of the new introduction to *Software Takes Command*, which is available for download from: http://manovich.net/DOCS/Manovich.Cultural_Software.2011.pdf.

That all disciplines which [*sic*] deal with contemporary society and culture – architecture, design, art criticism, sociology, political science, humanities, science and technology studies, and so on – need to account for the role of software and its effects in whatever subjects they investigate (2011: 7).

As software becomes increasingly ubiquitous, affects all kinds of cultural and societal processes and has a profound effect on how people live their everyday lives, software, so it is claimed, must be taken seriously as a site for understanding new social realities and transformations (see Mackenzie, 2006; Manovich, 2008; Fuller, 2008). My reasons for studying software stem from these realisations. I believe software must be taken seriously as an object of study that is permeated with the same kind of meanings, materialities, representations, imaginings, emotions, discourses, identities, and power relations as any other cultural object and formation. As with any new or nascent field of study, there is no established way to engage in software studies in ‘the right way’. On a broader level, therefore, this dissertation can be seen as one possible way of taking software seriously, of dealing with contemporary media culture through an account of software that focuses on the protocological and algorithmic level.

In historical terms, 2006 marked the year of the first organised attempt to establish the field of software studies more formally, with a workshop organised by Matthew Fuller at the Piet Zwart Institute in Rotterdam. However it was not until 2008 that software studies emerged in a more institutionalised form. First, a new research initiative on software studies was established at the University of California, San Diego, directed by Manovich. There was also a foundational workshop organised by the Software Studies Initiative in San Diego, and a new MIT book series on software studies was founded along with the publication of the book *Software Studies: A Lexicon*, edited by Fuller.

Framed as inherently interdisciplinary in scope, software studies ‘seeks to create an expanded understanding of code that extends significantly beyond the technical. It offers cultural and theoretical critiques to how the world itself is captured within code in terms of algorithmic potential and formal data descriptions’ (Dodge, Kitchin, and Zook, 2009: 1285). As such, existing research has examined software in terms of its mechanisms and expressive capacities (Kirschenbaum, 2008; Wardrip-Fruin, 2009), philosophical, ideological, and performative underpinnings (Berry, 2011; Chun, 2006; Galloway, 2006; Mackenzie, 2006), and the effects that software has on cultural production and everyday life (Fuller, 2003; Kitchin and Dodge, 2011; Manovich,

2008).

While much software studies research is related in some way to media studies departments, other academic fields have also made important contributions to bringing issues of software out of computer science departments. Most notable in this regard is some of the literature emerging from various geography departments in the UK. For the past decade or so, geographers and social theorists have examined how software can be said to transform and produce spatiality (Thrift and French, 2002; Dodge and Kitchin, 2004; 2005; Graham, 2005; Uprichard et al., 2009). These authors argue that space is automatically produced; everything from air travel and traffic lights to security systems run on and are operated by various codes and algorithms.¹¹ Digital networks have the capacity to generate ‘software-sorted geographies’ (Graham, 2005), but also create space itself, by providing the conditions for its effective operations (Dodge, Kitchin and Zook, 2009).

Software studies as a field is continuously growing. In the course of this three-year research period, software studies has established itself as a field in the fullest academic sense of the term. This includes the founding of a book series and a discipline-specific journal; academic courses, although no full-length degrees; workshops and conferences; and tenure-track openings where expertise in software studies is specified as a qualification.¹² Academic interest in software from a humanities and social science perspective only seems to be growing, as evidenced by the emergence of such related fields as platform studies and critical code studies, as well as by the growing interest and institutional stabilisation of what is now called

¹¹ As Thrift and French note: ‘Increasingly, spaces like cities – where most software is gathered and has its effects – are being run by mechanical writing, are being beckoned into existence by code’ (2002: 311).

¹² See MIT Press book series on software studies:

<http://mitpress.mit.edu/catalog/browse/browse.asp?type=6&serid=179>. Computational Culture (<http://computationalculture.net>) is the new journal of software studies. Its editorial group includes Matthew Fuller, Andrew Goffey, Olga Goriunova, Graham Harwood, and Adrian Mackenzie. ‘The journal’s primary aim is to examine the ways in which software undergrids and formulates contemporary life’ (from the about section). The first issue was published in the fall of 2011. Courses include ‘Software studies’ at the University of California, Santa Cruz (<http://ic.ucsc.edu/~wsack/fdm225/winter2010/schedule.html>), ‘Software studies’ at Goldsmiths University, ‘Code, software and serious games’ at Brown University (<http://thefollowingphrases.com/gamesSyllabus2.pdf>), ‘Coding culture’ at Utrecht University (<http://mtschaefer.net/entry/coding-culture-2010>), and ‘Politics of code’ at New York University (http://cultureandcommunication.org/galloway/2010fall-Politics_of_Code_syllabus.pdf).

digital humanities. That said, software studies is still very much in the margins of academia. There is disagreement about whether there really is a need for a demarcated field of study called software studies, or whether efforts should be put into making knowledge about software an integral part of already-established fields such as media and communication. In my view, software studies provides a very useful and highly-needed perspective on how to view media studies in times of ‘computational culture’.

Software studies and social media

While academic research on the productive power of software in configuring sociality in social networking sites remains scarce, there are a few notable exceptions. In the following, I will present some of the research that has been especially important in developing my own work. One particularly important source of inspiration is the scholarly work associated with the *Infoscape Research Lab* at Ryerson University, Toronto (Elmer, 2004; Langlois, 2008; Langlois et al., 2009a; 2009b; Langlois and Elmer, 2009; McKelvey, 2010; 2011). In particular, I am inspired by the doctoral dissertation of associate lab director Ganaele Langlois (2008), and the work that explicitly addresses social media from a software studies and ‘code politics’ perspective (Langlois et al., 2009a; 2009b). In her dissertation *The TechnoCultural Dimensions of Meaning* (2008), Langlois provides a critical analysis of media software (Amazon and MediaWiki), addressing power relations online as articulations between software, meaning, and subjectivation. While Langlois’ focus on aspects of meaning-making on the Web as understood from a Guattarian mixed-semiotics perspective falls outside the scope of my project, the ways in which she approaches technocultural subjectivation with a sensitivity towards issues of software provides an important foundation for discussing social networking sites in similar terms. Langlois sees software as a key actor in the shaping of sociality and subjectivity, where ‘different kinds of meanings are shaped and formed, from the symbolic meanings created through the interface to the cultural meanings that give form to software itself and define the practices of users’ (2008: 247). In similar terms, I view software as an important actor and participant in governing forms of sociality in the context of Facebook and Twitter.

More generally, my research project responds to a number of calls made by media scholars in recent years for a better understanding of software in studies of social

media. David Beer's article 'Power through the algorithm' (2009) constitutes one such influential call. As Beer contends:

Considering how popular Web 2.0 applications like MySpace, Facebook, Youtube, Delicious, and Flickr have become, there is a pressing need to explore with some detail this vision of power through the algorithm operating in their incorporation into users' lives [...] As things stand we simply do not understand how the material infrastructures of Web 2.0 play out in the lives of individual users, how the software constrains and enables, how it formulates hierarchies, shapes the things people encounter, and so on (2009: 999-1000).

Similarly, Niederer and van Dijck argue that 'non-human actors and coded protocols are often overlooked in the many optimistic Web 2.0 theories', and call for a heightened critical understanding and interrogation of the 'sociotechnical system that lies at the core of Web 2.0 platforms' (2010: 1384). What is needed, they argue, are analytical skills that take the medium-specific nature of the Web into account when attempting to understand how the Internet works. In this dissertation, therefore, in part motivated by these and similar calls, the aim is to examine some of the many ways in which the material infrastructures of Web 2.0 articulate power through software, thereby shaping specific ways of relating to self and others online. In doing so, this dissertation is an attempt to fill what Langlois et al. identify as a knowledge gap; namely, the 'need to examine how diverse elements and actors (human and non-human) are mobilized and articulated in specific ways' in order to shape sociality online (2009a: 416-417).

Medium theory and the materiality of media

From a media studies perspective, my interest in software as a structural force that shapes sociality is clearly indebted to the branch of media studies that has traditionally been concerned with the materiality of the medium as opposed to its content. The intellectual trajectory I refer to here began with the Toronto School of 'medium theory' (i.e. Innis, 1951; McLuhan, 1994; Ong, 2002; Meyrowitz, 1998) and its influence on German media theory and especially the writings of Friedrich Kittler (1999), and extends to more recent efforts from literary theorists who critique the hermeneutic tradition of interpretation, and argue instead for a focus on the 'materiality of communication' (i.e. Hayles, 1999; Gumbrecht, 2004).

The relevance of this trajectory for this dissertation pertains to its focus on theorising the specificity of media, and its view that the technical apparatuses of media influence social and cultural changes. The Canadian media theorists of the 1950-1980s, especially Harold Innis and Marshall McLuhan, are still regarded as the major reference for directions in media studies called ‘medium theory’. Medium theory seeks to understand how technology, medium-specific properties, and communication infrastructures shape and influence social and cultural transformations. While Innis, originally an economics historian, only produced two books having to do with human communication, the importance of the ways in which he pioneered the understanding of how knowledge is formed through the specificities of communication forms (i.e. writing and orality) cannot be overstated. Innis and McLuhan argue that media technologies are far from neutral carriers of information and communication. Rather, media are inherently ‘biased’. While Innis tied the inherent bias of media primarily to issues of time and space, McLuhan (1994) differentiated bias in terms of what he called ‘hot’ and ‘cool’ media, depending on the degree of participation they evoked in the audience. McLuhan is perhaps best known for his many aphorisms that still linger strongly in many writings and teaching on media and communication, of which ‘the medium is the message’ elaborated on in *Understanding Media* (1994 [1964]) is arguably the best-known.

McLuhan believed that one should favour the form of the medium over its content, as the medium influences how the message is perceived. Because the content can be so diverse, McLuhan regarded it as ineffectual in shaping human thought; ‘indeed, it is only too typical that the "content" of any medium blinds us to the character of the medium’ (McLuhan, 1994: 9). Importantly, the medium, which in McLuhan’s sense is to be understood broadly as a human prosthesis or as extensions of the central nervous system, encompasses everything that has the capacity to ‘change the scale or pace or pattern that it introduces into human affairs’ (McLuhan, 1994: 8). In other words, the medium in McLuhan’s sense may encompass everything from railroads to light bulbs. In the bestseller *The Medium is the Massage* (McLuhan and Fiore, 1967), McLuhan used the notion of massage to highlight how the specificity of a medium lies in the ways in which it impacts the human senses, as in massaging or making us perceive things in a certain way. The importance of McLuhan’s insights to my understanding of software – and indeed for software studies in general – pertains to the ways in

which the medium is understood in terms of its effects on sense perception and ways of knowing the world. In addition, as Mitchell and Hansen point out, ‘understanding media does not mean just understanding the individual mediums, but rather something like understanding from the perspective of media’ (2010: xi). This is highly relevant to my study as well, as I attempt to understand the ways in which sociality is formed on and through social networking sites from the perspective of the software. In other words, drawing on a medium-specific approach such as McLuhan’s does not so much entail a detailed examination of the properties of the medium, but rather implies a sensitivity towards the ways in which the medium induces specific changes and ways of relating to the world in general.

This concern with modes of accessing and knowing the world through material means has been a key theme in more recent post-humanist approaches that have focused on the *materiality of communication*. In his book *the Production of Presence* (2004), Gumbrecht suggests that media produce specific presences, as in possessing the power to present what is before our eyes.¹³ Similar to McLuhan’s thesis, the capacity to impact on the level of experience by constituting the conditions of appearance becomes an important theoretical point in investigating the ways in which software can be said to produce conditions for the intelligible and sensible.

The notion of the medium as the message or ‘massage’ has also been a direct inspiration for the kind of ‘media materialism’ developed by Friedrich Kittler. In contrast to McLuhan, however, Kittler did not seek to understand media, but rather to understand the ‘historical conditions of their emergence and the structures of communication and understanding they subsequently make possible’ (Gane, 2005: 28-29). Kittler drew inspiration for his ‘media theory’ from McLuhan, Lacanian psychoanalytic theory, the information theory of Shannon and Weaver and Foucault’s work on discourse.¹⁴ While Foucault focused on the ways in which institutions form

¹³ This point is also crucial in Martin Seel’s (2000) reflections on the ‘aesthetics of appearance’, understood as the condition in which the world is given to us through the production of a way of appearing, enabled by specific devices and institutions.

¹⁴ To call Kittler’s work ‘media theory’ is to some extent a paradox, as Kittler did not see himself as a media scholar. Kittler even pronounced the death of media with the advent of the computer and digital technologies, as ‘the general digitisation of channels and information erases the differences among individual media’ (Kittler, 1999: 1). In Kittler’s view, then, the concept of the medium loses its significance in connection to the computer, as everything merely converges into optical fibre networks.

particular ways of communicating and knowing the world, Kittler expanded the meaning-making capacities of institutions into the realm of technology. In his work *Discourse Network 1800/1900* (1990) especially, Kittler draws up a genealogy reminiscent of Foucault, showing how different technologies transformed communicative practices and knowledge in profound ways. For instance, Kittler used the example of Nietzsche's typewriter to show how 'technology transformed the physical connection of the writer to the text', thereby not only changing how writing was suddenly automated, but also altering the 'materiality of the text itself by organizing writing spatially through the distribution of discrete rather than continuous (as in handwriting) signs' (Gane, 2005: 30). It is in this way, through the specific technological capacities of storage, transmission, and processing, that the 'media determine our situation' (Kittler, 1999). As Mitchell and Hansen suggest, this radical proposal should be read more akin to 'determination' in a Marxian manner, as the infrastructure of capital and its influence on the social actor (2010: xxi). Seen in this way, media technologies 'form the infrastructural basis for experience and understanding' (Mitchell and Hansen, 2010: vii).

The importance of Kittler's media materialism in this context is the importance of grounding any analysis of media in a description of technological forms. Despite Kittler's hardware determinism and his assertion that 'there is no software' (1997), his concern with control and various 'instances of disciplining, inscription and programming, which, tied to varying discursive and technological regimes shape human and machine subjects' (Winthrop-Young and Gane, 2006: 9) are important analytical points. The usefulness of Kittler's 'radical post-humanism' (Gane, 2005) lies in its emphasis on the technical characteristics of the medium and the ways in which power is seen to be embedded within the materiality of communication. Contrary to Kittler's commitment to hardware and the subsequent abandonment of the notion of the medium in the age of computation, I do not regard the fact that everything might eventually be reduced to something else (binary code and voltage differences) as an argument against the study of software or its specificity. In contrast to McLuhan, who saw media as devices to enhance the senses, Kittler's "media" are first and foremost cultural techniques that allow one to select, store, and produce data and signals' (Krämer, 2006: 93). The difference here lies in the fact that Kittler

regards the media as technological a priori, producing the Lacanian real,¹⁵ rather than merely mediating the symbolic order, as McLuhan seems to suggest.

In this dissertation, however, it is not Kittler's engagement with Lacan that is of most importance; rather, I will draw on his extension of Foucault's approach to discourse. While Foucault failed to develop a rigorous view of the crucial role of technology in the production of discourse, Kittler felt the technical capacities of the medium were central to understanding the condition for meaning production. Importantly, Kittler did not merely study the technical characteristic of a medium, but rather combined it with historical and textual analysis. Discourse networks is defined as 'the network of technologies and institutions that allow a given culture to select, store, and process relevant data' (Kittler, 1990: 369). In Kittler's view, then, the media are essentially understood as discourse networks. As Jeremy Packer notes in an interview with John Durham Peters:

I'm much attracted to Kittler's Foucauldian tale on communication technologies and media as producing the brute facticity of discourse. Particular media or discourse networks allow certain statements, whether they be data, sounds, images, language, etc. to literally be made or not made. I think that's probably the most profound extension of Foucauldian thought that I've seen in terms of communication and media scholarship (Peters and Packer, 2012: 43).

The shift from discourse analysis to discourse networks implies an analysis of the morphology of technical processes, people, texts, and institutions that seek to 'delineate the apparatuses of power' (Kittler, 1990: xii). As Langlois pointedly observes:

¹⁵ Kittler adopts the notion of the 'real' from Lacan, as that which resists symbolisation. Lacan operates with three different orders: the real, the imaginary, and the symbolic. The real is that which exists before or beyond the mediation of language. The symbolic order is essentially the linguistic dimension, what the subject enters into after having acquired language. This is the realm of culture, always mediated by a third term. The symbolic creates reality by cancelling out the real. Reality is that which is named by language and thus can be talked about. What cannot be said in language is not a part of reality; it does not exist. Therefore, the real does not exist. See for instance Bruce Fink's 'The Lacanian Subject' (1995) for a good overview on Lacan's psychoanalytic theory. Kittler uses this to make a point about positing media as apparatuses that resist symbolisation, thereby having the capacity to record, store, and produce the real. As Krämer points out: 'The significant point here is that analog, technological media are the first to record events that transpire outside of the audible and visible realms. The real itself is saved by the phonograph, by photography, and by cinematography, it is transmitted by radio and television, and it is – at least in part – also even produced' (2006: 101).

Throughout numerous analyses of specific texts produced through different media, Kittler expands the concept of mediality through a detailed analysis of the traces of specific media present in the texts being analyzed. The text, then, becomes a valuable tool for defining the characteristics of a medium – characteristics that are not only aesthetic, or cultural, but also experiential. The analysis of a discourse network, including the texts produced by that network, allows for a critical reflection on the genealogies of media systems (2008: 65)

In this dissertation, following this kind of media analysis proposed by Kittler (as inspired by Foucault) allows the medium to be taken seriously as a material-discursive practice. Discourse is not seen as the opposite of materiality. Rather, discourse is material and the material is discursive. The material-discursive is here understood akin to what Karen Barad proposes in her account of agential realism (2003; 2007). Following Foucault's notion of discourse, Barad warns against making the representational mistake of equating discourse simply with language or human speech. Rather, she argues that '[d]iscourse is not what is said; it is that which constrains and enables what can be said' (2003: 819). The point, as Barad suggests, 'is not merely that there are important material factors in addition to discursive ones; rather, the issue is the conjoined material-discursive nature of constraints, conditions, and practices' (2003: 823). It is not that there is discourse on the one hand and materiality on the other. 'Discursive practices and material phenomena do not stand in a relationship of externality to one another; rather, the material and the discursive are mutually implicated in the dynamics of intra-activity' (Barad, 2003: 822).

In terms of 'material media studies', I am particularly influenced by the work of Alexander Galloway (2004; 2006) and his material understanding and explorations of digital technologies and infrastructures. In his book *Protocol* (2004), Galloway engages in close readings of how the Internet protocol functions, in order to make an argument about the transformation of control from centralised to decentralised networks. In describing his methods, he states that he attempts to 'study computers as André Bazin studied film or Roland Barthes studied the striptease: to look at a material technology and analyze its specific formal functions and dysfunctions' (Galloway, 2004: 18). The emphasis on protocols as articulations of power becomes an important case in point when considering social networking sites as well, where new types of centralised protocols have become powerful actors that shape and configure sociality, the nature of interaction, and circulation of information.

Process-relational philosophy

Theoretically, this project is inspired by theories that in a recent conference call were lumped together under the umbrella term ‘nonhuman turn’.¹⁶ Rather than taking this alleged turn at face value, the conference call encourages the naming of the intellectual and theoretical tendencies in relation to the importance of theorising objects, things, relations, intensities, affect, processes, and nature within the humanities and social sciences. Rather than dismissing the importance of the human aspect, I believe theories that emphasise process-relational and nonhumans offer a useful way to situate this study of power through software. Thus, in an age where software ‘represent[s], collate[s], sort[s], categorize[s], match[s], profile[s] and regulate[s] people, processes and places’ (Kitchin and Dodge, 2011: 10), theoretical perspectives that might account for media as processual relations are needed.

In the midst of ‘algorithmic culture’, theoretical debates and discourses that seek to address the changing notion of the object and the mediatic have flourished (see for instance Parikka, 2011a for a useful overview). While some of the theoretical labels given to these academic debates are novel (i.e. object-oriented philosophy¹⁷ and speculative realism¹⁸), most of the theoretical legacies addressed here relates to what can be considered a *process-relational* trajectory, which includes thinkers such as Whitehead, Deleuze, Serres, and Latour, among others.

While I will not engage in a metaphysical debate about the nature of objects, it is worth noting how important the notion of the ‘object’ has become in a wide range of academic disciplines, where objects are seen to exist separate from human subjective experience. In what is often referred to as the ‘material turn’ in cultural studies, media studies, sociology, and anthropology, especially since the late 1980s, objects or things have gained momentum as a matter of concern in and of themselves. Perhaps more so

¹⁶ <http://www4.uwm.edu/c21/pages/events/conferences.html> (accessed February 9, 2012)

¹⁷ For a useful discussion of the status of the object, the recent debates in ‘object-oriented ontology’ (OOO), and ‘object-oriented philosophy’ are quite instructive. Proponents argue that objects have an existence beyond the relations they forge. See for instance Levi R. Bryant’s blog ‘Larval Subject’s’ (<http://larvalsubjects.wordpress.com>). Graham Harman (who coined the term ‘object-oriented philosophy’) has written a book about Latour and objects called *Prince of Networks*. He too blogs, at <http://doctorzamalek2.wordpress.com>, another useful resource.

¹⁸ See my blog post on the topic from October 27, 2010: <http://tainabucher.com/?p=225>.

than anyone else at that time, Arjun Appadurai in *The social life of things: Commodities in a cultural perspective* (1986) argued that things do not merely signify or are suggestive of things by virtue of having meaning transposed onto them by human signification. Rather, things also have the power to signify, suggest, and make sense in their own right by virtue of the stories they tell about a social context. While things and objects have been a matter of concern long before ‘material cultural studies’, especially in theories concerning commodities and fetishism after Marx, for our current purposes the ways in which the nonhuman has been conceptualised in science and technology studies (STS) and Actor-Network Theory (ANT) provide a first helpful theoretical entry point into situating the study of software and sociality.

Actor-Network Theory: Objects, relations, and agency

Actor-Network Theory was developed in the 1980s as an ‘ethno-methodology’ of close readings of scientific practices as articulations of heterogeneous networks (see Callon, 1986; Latour, 1988; Law, 1992). As Latour points out, ‘[i]t was at this point that nonhumans - microbes, scallops, rocks, and ships - presented themselves to social theory in a new way’ (2005: 10). ANT argues for a radical symmetry between human and nonhuman actors, and sees the social and technical not as separate entities that can be considered independently of each other, but rather as engaging in symbiotic relationships organised in networks. As such, agency is not restricted to humans. Rather, ‘any thing that does modify a state of affairs by making a difference is an actor’; one needs simply to ask whether something ‘makes a difference in the course of some other agent’s action or not’ (Latour, 2005: 71). The classical ANT view on objects is that ‘they do not exist “in themselves” but are the effect of a performative stabilization of relational networks’ (Pels et al., 2002: 11).¹⁹ In Latour’s words, objects are ‘hybrids’ that signify a break with the dualist paradigm of viewing an entity as either social or technical. Latour’s gun metaphor is exemplary in this regard.

¹⁹ As John Law explains, this way of understanding objects in terms of relations is derived from semiotics, where the significance of a term depends on its relations (2002). The meaning of an object in this account, then, must be understood as a relational effect. Law provides the empirical example of the Iberian maritime technology, a network that consists of a temporarily stable set of objects that stand in relation to each other. An object ‘remains an object while everything stays in place and the relations between it and its neighbouring entities hold steady’ (Law 2002: 93).

According to Latour, it cannot be either ‘guns kill people’ or ‘people kill people; not guns’ (1994: 30). Rather, what needs to be accounted for is the agency of the combination between the gun and the person. Latour elaborates in the following manner:

What does the gun add to the shooting? In the materialist account, everything: an innocent citizen becomes a criminal by virtue of the gun in her hand. The gun enables of course, but also instructs, even pulls the trigger – and who, with a knife in her hand, has not wanted to stab someone or something? Each artifact has its script, its “affordance”, its potential to take hold of passersby and force them to play roles in its story. By contrast, the sociological version of the NRA renders the gun a neutral carrier of will that adds nothing to the action, playing the role of an electrical conductor, good and evil flowing through it effortlessly (Latour, 1994: 31).

Who then is to be considered the actor in this case? ‘The gun, the shooter? No, it is something else (a citizen-gun, a gun-citizen) – a hybrid actor’ (Latour 1994: 32). According to Michael, ‘what Latour aims to do is to show how the new hybrid entails new associations, new goals, new translations and so on. As one enters into an association with a gun, both citizen and gun become different’ (2004: 9). Objects in this sense are ‘enactments of strategies, and actively participate in the making and holding together of social relations’ (Pels et al., 2002: 11). Crucially, objects in this relational account are not very stable, despite the fact that they possess temporal stability. Any new circumstance forges new relations, thereby destabilising the object while simultaneously generating a new momentary association or gathering, which in turn gives the object a new shape.²⁰ The relational view on objects in the accounts of Latour and ANT is highly indebted to the process-relational philosophy of Albert Whitehead.

Whitehead: Process-relational philosophy

For Whitehead, objects (or ‘actual entities’) are combinations or clusters of relations. Actual entities are only knowable in their becoming, as opposed to their being. As Whitehead suggests, ‘how an actual entity becomes constitutes what that actual entity

²⁰ John Law and Annemarie Mol have introduced the concept of ‘fluid objects’ to account for the ways in which objects flow into new configurations rather than remain stationary (Mol and Law, 1994). A similar notion of fluidity in objects prefigures in Deleuze, who refers to the ‘continuous variation of matter’ as *objectile* (1993: 19). According to Deleuze, the objectile represents a new kind of object, no longer imbued with a beginning or an essence.

is. Its “being” is constituted by its “becoming”. This is the “principle” of process’ (1978: 23). As with the relational view of Latour, where heterogeneous relations constitute an object and always implicate other actors, Whitehead sees an actual entity as a compositional thing. Contrary to Whitehead, however, Latour rejects the notion of becoming, as there is nothing that would exceed individual actors (Harman, 2009: 101). Arguably, part of the reason Whitehead’s view of objects and process has become crucial lies in his break with the view of objects in terms of substance and essence. Indeed, “actual entities” are the final real thing of which the world is made up. There is no going behind actual entities to find anything more real’ (Whitehead 1978: 18). This has important consequences for the analytical treatment of software, as it removes the possibility of accessing an underlying truth.

Studying processes and heterogeneous objects themselves, however, is not exactly a straightforward undertaking. As Mike Michael point out: “Choices” have to be made as to what to include and exclude in its composition’ (2004: 10). Tracing associations and making decisions about *which* relations and *which* actors to include in the study of hybrids or actual entities, is ultimately an analytical fabrication. The point is not to account for ‘the empirical accuracy’ of the hybrid, but rather to employ a ‘heterogeneous perspectivism’, with which one seeks to find evidence of ordering and disordering from the varying perspectives of the relevant agencies involved (Michael, 2004: 10, 20). This is important when studying such heterogeneous objects as software, as it would be difficult, indeed impossible, to account for empirical accuracy. Rather than seeing software as a substance with some kind of essence (for instance its source code), understanding software in process-relational terms derived from Latour and Whitehead among others, offers a way to look for the various material-discursive associations that coalesce to produce new realities. As Whitehead points out, ‘there is nothing which floats into the world from nowhere’ (1978: 244)²¹.

²¹ In order to account for the ways in which actual entities ‘become’ out of something, he introduces the notion of *potentiality* and *prehensions*. ‘Becoming’ is always becoming concrete out of a multiplicity, as an “entity” – means nothing else than to be one of the ‘many’ which find their niches in each instance of concrescence’ (Whitehead, 1978: 211).

Prehension is a key concept in Whitehead’s metaphysics, and refers to the disparate data, including energies, emotion, purpose, causation, and valuation, that concrescence or combine to produce actual entities (Michael, 2004). For Whitehead, actual entities or occasions become concrete through a process he calls *concrescence*, that is, the ‘production of novel togetherness’ (1978: 21). This insistence of the becoming or thickening of an actual entity

Assemblage and the quasi-object

At the core of the nonhuman turn lies not only the importance of acknowledging the agential capacities of nonhumans, but also, as we have seen, the emphasis on relationality. Perhaps more so than any other concept, the notion of assemblage has served as way to account for the ways in which relations are assembled for different purposes. Deleuze and Parnet view assemblage as a ‘multiplicity which is made up of many heterogeneous terms and which establishes liaisons, relations between them’, where the only unity ‘is that of co- functioning: it is a symbiosis, a sympathy’ (2007: 69). The concept of assemblage usefully points towards the ways in which reality and its specific entities are above all compositions, where different relations are put together to work as a whole in between contingency and structure, organisation and change (Deleuze and Guattari, 1987). As Coonfield elaborates in his account of the Deleuze/Guattarian notion of assemblage, ‘a machinic assemblage is understood not as a thing, but as a process, an ongoing organizing of multiplicities, of relations between elements and forces, that produces affects’ (2006: 290). Seen in this way, assemblages emerge as an entity of connected relations.

What is important to point out is that software should not merely be thought of as an assemblage in the sense of gathering intact subjectivities and technical objects, but rather in the sense of its original French meaning of *agencement* –a process of assembling rather than a static arrangement (see Packer and Wiley, 2012; Callon, 2007). In this sense, the notion of *agencement* connects to my notion of programmed sociality as the ways in which actors are composed and articulated in and through specific coded means of assembling and organising. As Michel Callon points out, ‘agencement has the same root as agency: agencements are arrangements endowed with the capacity of acting in different ways depending on their configuration’ (2007: 320). Analytically, this implies a greater emphasis on the ways in which assemblages of heterogeneous elements are arranged, but also *in arrangement* (as in an ongoing assembling), in order to think about what is connected to what, and to what purpose.

Michael Serres’ notion of the quasi-object offers one particularly useful way to account for the relationality at work in an utterly sociotechnical world. For as Serres

from a multiplicity of possibilities (or potentiality) has had an enormous influence on Deleuze’s philosophy of the virtual.

proclaims, 'our relationships, social bonds, would be airy as clouds were there only contracts between subjects' (1995: 87). One needs operators that draw people together in particular relations. These operators or mediators are what Serres calls quasi-objects. Serres provides the example of the ball to illustrate his notion of the quasi-object as that which organises the relations that fluctuate in and around it.

A ball is not an ordinary object, for it is what it is only if a subject holds it. Over there, on the ground, it is nothing; it is stupid; it has no meaning, no function, and no value. Ball isn't played alone [...] The ball isn't there for the body; the exact contrary is true: the body is the object of the ball; the subject moves around this sun. Skill with the ball is recognized in the player who follows the ball and serves it instead of making it follow him and using it [...] Playing is nothing else but making oneself the attribute of the ball as a substance. The laws are written for it, defined relative to it, and we bend to these laws (Serres, 1982: 225-226).

The example of the ball shows how objects or quasi-objects should be seen as active participants in social relationships, rather than as passive end-points of human action. Indeed, the social context would not be possible without the intervention and 'actions' of objects. For Serres, then, the ball becomes a means to explain the sociotechnical, for the ball is not merely a social construction. The ball bends human practices in different ways, configuring and reconfiguring the relations around it. As Massumi points out, it is not the player that is the subject of the play; it is the ball (2002: 73). Therefore, the quasi-object also designates a quasi-subject. As the quote shows, the ball does not just lie there passively on the ground doing no work in the world. Quite the opposite: in subtle ways, the ball has the capacity to affect human beings, to make a difference to the state of affairs. Bodies are bent, giving the players a reason to run after it or move around it. The game itself becomes an emergent entity that organises itself around the different potentialities and intensities that coalesce around the ball as it directs the movement of the players. As Massumi puts it, '[t]he ball moves the players. *The player is the object of the ball*' (2002: 73). What a quasi-object does, then, is to bring relations together in constantly shifting configurations.

Following these theoretical views on the agency of objects,²² software can be seen as having agency in the sense that it might affect the course of action of other agents.

²² Latour provides the example of the speed bump and the doorknob to illustrate how things can have agency. Whereas the speed bump makes people slow down, the doorknob enables people to enter a room.

While software is sometimes seen to possess a full-blown agency, for example in discourse on artificial intelligence, more often software is understood to have some sort of ‘secondary agency, that is, supporting or extending the agency of some primary agency’ (Mackenzie, 2006: 8). In his book *Cutting Code*, Adrian Mackenzie concludes that there are two broad patterns at play in the agential relations that occur in software, one that has to do with ‘involution’ and the other with ‘convolution’ (2006: 182). Drawn from the vocabulary of Deleuze and Guattari, involution signifies the emergence of a ‘symbiotic field that allows assignable relations between disparate things to come into play’ (Ansell-Pearson, 1997: 130). In terms of software, then, agency lies in the fact that it ‘opens up to new possibilities and gives rise to diverse realities’ (Kitchin and Dodge, 2011: 38). Convolution on the other hand refers to the compositionality of software, and blurs the starting relation. The fact that many existing forces and relations are encompassed by software, in effect, blurs the agential starting points. This complicates the question of ‘who says or does what’ (Mackenzie, 2006: 182). Software can be viewed not merely as an object deprived of agency, but as being fundamentally imbued with agential capacities. This is perhaps the main argument for its importance as an object of study.

Power: A Foucauldian account

The fields of media and software studies are united by their concern with questions of power. When humanists and social scientists seek to understand the operations of society – what drives society, organises governments, determines knowledge production, and prompts social change – power is often the concept used to make sense of these phenomena. In many traditional social theoretical accounts, power is seen as a kind of repressive force. In one of the most influential of these, Max Weber defined power as the capacity of people to ‘realize their own will in a social action even against the resistance of others’ (Wallimann et al., 1977). Power in this sense is often described in terms of a ‘power-over’, often attached to social structures like class, ethnicity, and gender. Especially in terms of class, Marxist ideas of power have had a profound impact on how media scholars have come to view the hierarchical operations of power as connected to the domination of economy. Broadly stated, questions of power in a Marxist tradition have used concepts such as ideology and hegemony to express how certain beliefs are propagated by a ruling class, and

subsequently taken up by lower classes as part of their own belief system. These notions of power have been highly influential in relation to the political economy of the media, as they have been particularly well-suited to explain the asymmetries of power in capitalist societies. Especially in terms of mass communication, the question of how media influences its audience and readers by means of reproducing dominant institutional relationships or otherwise inflecting on people's opinions and behaviour has been at the fore in many related disciplines concerned with the power of the media.²³

In the context of this study, however, power is not understood as a repressive force. Rather, following Foucault, power is seen as *productive*. Foucault understood power as a relation between forces, not as a thing in and of itself.²⁴ As such, power does not exist. It persists only insofar as it emerges as part of a relation, when it is put to work. Power, in Foucault's terms, is always exercised, and may only be analysed as part of the structures put in place to guide the possibility of conduct. While power figures as a central concept throughout most of Foucault's work, the most pointed summary of the concept of power can be found in his article 'Subject and Power' (1982), where he writes:

In itself the exercise of power is not violence; nor is it a consent which, implicitly, is renewable. It is a total structure of actions brought to bear upon possible actions; it incites, it induces, it seduces, it makes easier or more difficult; in the extreme it constrains or forbids absolutely; it is nevertheless always a way of acting upon an acting subject or acting subjects by virtue of

²³ Theories of hierarchical and repressive power structures in media and communication research range from the critical-analytical traditions introduced by the Frankfurt School (i.e. Adorno and Horkheimer) that sought to explicate the conditions of belief. Some branches of cultural studies influenced by structuralism and Althusser's reformulation of the concept of ideology (i.e. Stuart Hall) have had a profound impact on addressing how media have a power to construct the real. Within more social science and political communication-oriented media studies, the concepts of 'agenda-setting' (i.e. McCombs and Shaw, 1972) and 'framing' (i.e. Etman, 1993), have been influential in theorising the power of media in a hierarchical manner. While the former designates the ways in which media influence *what* people think about, the latter accounts for *how* people think about certain issues. This, of course, is a caricature of how power has been theorised in terms of being understood as a repressive force, and does not really do justice to the complexity or socio-historical context of these theories. For example, agenda-setting research merely states that the media function to set an agenda, rather than actually changing or repressing certain attitudes.

²⁴ The empiricist John Locke already conceived of power in these relational terms. As Whitehead cites: 'fire has a power to melt gold [...] and gold has a power to be melted. Power thus considered is twofold; viz. as able to make, or able to receive [...] power includes in it some kind of relation' (Whitehead, 1978: 57–8).

their acting or being capable of action. A set of actions upon other actions (1982: 789).

The notion of power that Foucault has in mind is not so much a thing, then, but a capacity. It is the capacity of power exercised, the capacity of power relations to conduct and to lead. As Foucault exemplifies:

The exercise of power consists in guiding the possibility of conduct and putting in order the possible outcome. Basically power is less a confrontation between two adversaries or the linking of one to the other than a question of government. This word must be allowed the very broad meaning which it had in the sixteenth century. "Government" did not refer only to political structures or the management of states; rather it designates the way in which the conduct of individuals or groups might be directed (1982: 789-790).

What is important here is the dual logic of power. Precisely because power is first and foremost seen as a relation between forces, power operates in specific formations called *diagrams* that materialise in the 'distribution of the power to affect and the power to be affected' (Deleuze, 2006: 61). Power in Foucault's view cannot rightfully be conceived of as repressive, carrying the force of prohibition. Rather:

What makes power hold good, what makes it accepted, is simply the fact that it doesn't only weigh on us a force that says no, but that it traverses and produces things, it induces pleasures, forms knowledge, produces discourse' (Foucault, 1980: 119).

Because it is productive, power is necessarily also a constraining force. While it produces ways of knowing the world, power also limits alternative ways of being and talking. What is of interest in this study, however, is not so much the silences or ways of knowing that are ruled out, but the ways in which power is productive, how power induces and seduces. Following Foucault's later writings, it is not a question of *who* holds the power that should be of concern. Rather, the question is *how and in what ways, through which techniques and procedures the effects of power are produced*.

I believe that this two-fold nature of power to affect and to be affected, to enable and constrain, is the most useful way to investigate the nature of programmed sociality in such dynamic media environments as social networking sites. The difficulty with the relational account of power, however, is the absence of any location of power. Power is nowhere and everywhere at the same time, making it difficult, if not impossible, to locate power as if it were a thing. In order to get a better sense of how to analyse power relations, it might therefore be useful to go back to some of Foucault's earlier

genealogical writings (i.e. 1976; 1977). Here, he locates the operation of power on a concrete institutional and structural level (i.e. the hospital, the prison, the factory). By studying such institutional settings, Foucault discussed the ways in which power operates in an individuating fashion. In this sense, he did not for example use the figure of the prison as a form of power, but rather as a technique of power, a concrete architectural way of organising and managing subjects. The productivity of power is located within specific modern social apparatuses that actively work to fabricate society and modes of conduct (Gordon, 2001: xix). Power thus operates in the very practices, techniques, and arrangements of localised institutions.²⁵ As Foucault points out:

Power has its principle not so much in a person as in a certain concerted distribution of bodies, surfaces, lights, gazes; in an arrangement whose internal mechanisms produce the relation in which individuals are caught up (1977: 202).

The point for Foucault was precisely to leave the repressive hypothesis of power behind, and instead to look at how power relations are productive of the conditions of possibility. This does not mean that power is somehow ‘harmless’. To the contrary, the political and potentially harmful aspect of power lies in the apparent neutrality of the functioning of power. Power as such is much more ubiquitous than the thesis of repression would hold. Because power transforms, it transforms all the relations that form part of the diagram. Power is precisely a technique for the transformation of arrangements whereby different bodies are distributed and circulated in a network of relations (Foucault, 1977: 146).

Although Foucault focuses more on institutions and does not particularly touch upon technology as such, technology enters his lexicon in *Discipline and Punish* (Rajchman, 1988: 101). The operation of what in this book is called disciplinary power hinges on a set of mechanisms or technologies, including examination, observation, and judgement. Taking the architectural form of the prison as the starting point for his analysis of power, Foucault highlighted the ways in which certain spaces

²⁵ Foucault’s genealogical method is also known as ‘microphysics’. As Gordon writes in the introduction to the *Essential works of Foucault: Power*: ‘the microphysical emphasis of the seventies books was in part, an argument for the primacy of analyses of practices over analyses of institutions – explaining the origins of the prison, for example, on the basis of analysis of the changing meaning assigned to the practices of punishing’ (Faubion, 2001: xxv).

are designed or arranged to achieve a certain goal, permeated through and through by forces of power that are mutually co-constitutive. For my purpose, the organisational and governing mechanisms that function through the techniques of power provides a useful way to investigate the forms of programmed sociality that emerge in and through social networking sites. An emphasis on a diagrammatics of software allows for an analytical perspective that focuses on the specific techniques through which the ways of relating to self and others are produced online.²⁶ The particular usefulness of Foucault's architectural vision of the organisation and practices of power in the context of this study lies in the ways in which power can be located within specific technologies of government, where power relations become a question of studying arrangements and distributions of the sensible and knowable.

Towards a productive view of software studies

I began this chapter by stating that software is seen as a material-discursive phenomena that in a non-representational sense is first and foremost studied for what it can be said to effectuate. An engagement with software studies in the context of this dissertation therefore does not seek the what, but rather the how. The question, as Foucault put it, does not pertain to 'how' in the sense of 'how does it manifest itself?' but 'by what means is it exercised?' (1982: 786). Here, I rely on software studies to understand the ways in which sociality is mobilised and articulated through software.

The theoretical trajectories presented in this chapter provide a way to view software in a manner that departs from the classic transmission model of communication. Instead of addressing software in terms of a causal sender-transmission-receiver model, viewing software as actants, actual entities, *agencement*, and quasi-objects allows for the interrelations and articulations of the machinic and the enuncitative. Here, Karen Barad's conception of the material-discursive offers a useful way to understand the mutual constitution and inseparability between the discursive and the material. Rather than merely viewing the discursive as a synonym for language, the material-discursive signifies the ways in which the conditions for the intelligible and sensible are always

²⁶ See Rodowick, 1990; Elmer, 2003; Parikka, 2011b, for useful discussions on the notion of diagrammatics. Parikka for example writes about an 'operative diagrammatics' to the study of media, where the objects or media technologies themselves are analysed for their inscriptions and ultimately for the ways in which they can be said to distribute power.

socio-historically grounded in specific material supports, or rather, material (re)configurations. As she asserts:

Discursive practices and material phenomena do not stand in a relationship of externality to each other; rather, the material and the discursive are mutually implicated in the dynamics of intra-activity. The relationship between the material and the discursive is one of mutual entailment (Barad, 2007: 152).

While my use of Barad's notion of the material-discursive does not really do justice to her broader agential realism argument, I think her conception summarises nicely what both Foucault and later Kittler described in terms of the materiality of regulatory and discursive practices and 'discourse networks', respectively. This dissertation combines the Kittlerian concern with the operation of media, with the more McLuhanesque concern of how media create sensoriums. It is important to point out that I do not seek to fully adopt the theories of McLuhan nor Kittler, but rather use some of their insights while disregarding others. From McLuhan I adopt his notion of medium specificity connected to the ways in which different media forms have the capacity to produce certain sensoriums or conditions for the sensible and intelligible. From Kittler, I adopt the assumption that media technologies produce specific regimes of power, and that one useful way to begin an analysis of this power goes via the workings of specific technologies, in order to uncover the 'systems of rules that govern its functioning in the first place' (Gane, 2006: 78). Thus, the kind of media materialism I invoke here pertains to Kittler's method of 'description and analysis of technological forms' (Gane, 2005: 29), used to address the powerful infrastructural role of media technologies.

Importantly, the theoretical foundations and conceptual frameworks discussed throughout the chapter emphasise the importance of paying attention to the question of *how* media/software operate and produce the conditions for what is intelligible and sensible. It implies an intermingling of different types of actors, where software needs to be viewed as a gathering of various heterogeneous relations. Social networking software combines not only an amalgam of different technologies and infrastructural orderings such as protocols, algorithms, and code; software, as with Serres' ball, brings actors together and organises relationships in particular ways. All these accounts have in common a desire to overcome some otherwise deeply-entrenched dichotomies, such as object-subject or nature-culture, and point instead to the multi-causal relations of the world. Moreover, these theories help explain the ways in which

software moves, circulates, and assembles. The emphasis on the processual and relational allows a shift in focus, away from an ontological commitment to what software is, to the question of what software is capable of doing. In writing about the performativity of code, Mackenzie for instance argues that the Linux operating system should be understood as an operational object that ‘coordinates the encounters of specific social actions pertaining to information and communication networks’ (2006: 76). In a similar vein, software assemblages such as Facebook and Twitter should be understood as operational objects that combine heterogeneous relations in a continuous and emergent way, configuring and reconfiguring the specific arrangements of which these relations are a part.

This means software can be seen as an actor that has the capacity to modify a state of affairs by making a difference. These agential capacities of software clearly point to the concept of power. Relying on Foucault’s understanding of power posits software not as a deterministic force, but rather as a ‘force relation’. Understanding how power articulates through software thus implies a sensibility towards the ways in which software in the broadest sense pushes, urges, and compels other actors to do something. It might be important to point out that using a Foucauldian understanding of power does not mean using the term in one single way, as if there were something called Power with a capital P. Rather, the usefulness of Foucault’s concept of power in understanding the relations between software and subject as it plays out in social networking sites, stems from at least two important characteristics: 1) power is above all productive, and 2) the notion that power operates and can be exercised through specific techniques and procedures, architectural forms and technologies.

Before we can continue to study the technical structures of software-mediated processes, we need to gain a better understanding of the nature of software itself. As Manovich importantly reminds us, ‘if we don’t address software itself, we are in danger of always dealing only with its effects rather than the causes: the output that appears on a computer screen rather than the programs and social cultures that produce these outputs’ (2011: 3). In the next chapter, I therefore explore the multifaceted nature of software by engaging with some of the previous work in software studies and related fields.

Chapter 3. Perspectives on software

The stuff we call “software” is not like anything that human society is used to thinking about. Software is something like a machine, and something like mathematics, and something like language, and something like thought, and art, and information...but software is not in fact any of those other things. The protean quality of software is one of the greatest sources of its fascination. It also makes software very powerful, very subtle, very unpredictable, and very risky (Bruce Sterling quoted in Galloway 2004: 166).

Software is a term that is easy to use but very hard to explain. Most people have an intuitive understanding of what software is. It is often defined with reference to an example, such as that of a particular program, like Microsoft Word. Dictionary definitions, being more general, emphasise how software is this thing that tells the computer what to do, where the existence of software can be formally traced to a textually-scripted source code. Perhaps were we to leave it at this, software would not seem that difficult to comprehend, nor that interesting to study. But source code alone cannot be credited for making the Word program show up on users’ computer screens, and once we start asking questions about the nature and ways of software, the issue proves to be more complicated. What exactly is doing the instructing? What are the necessary elements that need to be in place for something to be called software? When, where, and how is software?

While this chapter will not provide answers to the metaphysics of software, I suggest that it is exactly this type of question that is important to ask if we are to adequately understand the material-discursive condition of social networking sites. If software is simply analysed as instructions that tell a computer what to do, it is not enough to say that social networking sites are software. If we are to appreciate and address the questions asked in the beginning of this dissertation, what is needed is a better understanding of the multidimensionality, operational logic, and variegated nature of software.

In the following account, I will follow Adrian Mackenzie and his belief that software has a ‘variable ontology’, that ‘suggests that the essential nature of the entity is unstable’ (2006: 96). The variable ontology of software means that ‘questions of when and where it is social or technical, material or semiotic cannot be conclusively answered’ (ibid.). While there are good reasons to be concerned with the question of

what software is, here it provides a better basis for understanding what software can do, or at least what it is capable of doing in principle.

Thus, before one can analyse the ways in which software constitutes a powerful force in social networking sites, there is a need to be more explicit about software itself. One of the core tasks of software studies, in my view, is precisely to address software as a concept. If the humanities and social sciences have something important to offer to the study of software, it lies in the analytical and critical attitude so deeply entrenched in these fields. The theoretical and methodological tools of these sciences – disciplines that focus on meaning and culture – can complement the structural formalism of computer science with a view of software that highlights its existence as a symbolic and cultural formation.

Such critical work is already well underway (see Chun, 2011a; Fuller, 2003; Mackenzie, 2006; Manovich, 2008). In this chapter, I draw on this body of literature in order to be better equipped to understand the ways in which software organises and arranges sociality on social networking sites.

It is important, however, to remember that the aim of such a review is not to end up with a definitive account of the nature of software, or to define what software is, in and of itself. The ontological questions with which I am concerned are first and foremost a means by which to address *the epistemological issue*; namely, how can we as media scholars study and understand software fruitfully and productively?

In order to appreciate *what* software does, I take it to be necessary to first say something about the multiple ways in which it exists. I will do this by focusing on four different levels, or conceptualisations: software as code, algorithms, execution, and practice. I treat these as necessary conditions of software, but I do not make the stronger claim that they are necessary *and sufficient*. I do not suggest that software can be reduced to these four levels. I do claim, though, that these four elements of software are either salient or important enough to merit a discussion and review of what they are, and of how and indeed whether they can be studied. An analysis of software as code, software as algorithm, software as execution, and software as practice thus makes up the conceptual foundation for the chapters to come.

Software: A shifting nexus of relations

Software is essential to the functioning of new media. It is deeply embedded in the systems and infrastructures that make up today's media environments. As such, software encodes the world in important ways. Yet, software is often understood from a purely instrumental perspective, as a stable tool that can be used to accomplish certain tasks. Seen in this way, merely as a functional entity, as a set of instructions telling a machine what to do, it is perhaps not surprising that software has not been at the centre of attention of the social sciences and humanities. After all, as Wendy Chun points out, even the common-sense dictionary definition of software has reduced it to a set of instructions. She goes on to point out the problem with such definitions:

[...] which treat programs and procedures interchangeably, erase the difference between human readable code, its machine readable interpretation, and its execution. The implication is thus: execution does not matter - like in conceptual art, it is a perfunctory affair; what really matters is the source code (2011b: 100).

What is needed, according to Chun, is a distinction between the different layers and functional aspects of software. Neither the source code nor its executable layer is software's essence. Seen in this way, software does not merely instruct machines, but also the conduct of those who use the machines. This is not to say, however, that software instructs in a deterministic way. By means of its execution, software becomes suggestive of things, in that it enables and constrains certain kinds of action. In this sense, software can be understood in terms of the concept *affordance*.

The perceptual psychologist J. J. Gibson (1986) originally coined the term *affordance* to designate a way to describe the relational dynamics between animals and the environment. According to Gibson, 'the *affordances* of the environment are what it *offers* the animal, what it *provides* or *furnishes*, either for good or ill' (1986: 127). *Affordance* is not an inherent property of the environment, but emerges relative to the animal. For example, while the cave 'means' shelter for one type of animal, it might be perceived as a trap for another. Depending on the context, then, the environment shifts meaning accordingly. Similarly, we could say that software 'means' different things, depending on the context. The concept of *affordance* usefully highlights how the materiality of the environment (whether natural or software-mediated) can be understood in terms of how it signifies possibilities for action rather than determines actions.

Here I adopt what can be seen as an expanded view of software, where software needs to be seen, following Adrian Mackenzie, as a ‘shifting nexus of relations, forms and practices’ (2006: 19). Paradoxically, such an expanded view does not make software more comprehensible, in that it can be seen to lead an *event-like* existence, actualising in multiple ways, thereby retaining an openness to reinvention (see Fraser, 2006: 130). This however does not imply that software needs to be seen as entirely novel in its constitution and operation. An understanding of software will still benefit from an account of both the conditions that prevail, and the novelty of its composite nature. This chapter is therefore concerned with some of the prevailing conditions of software, and asks how these conditions can be studied and understood as the impetus for new forms and constellations.

Code

Code, or source code, is the very backbone of software, given that software runs on code. Code is ‘usually understood as the concretisation of general algorithms instantiated into particular programming languages in plain text files’ (Berry, 2009). Source code reads as a description of a process, a sequence of commands telling the computer what to do. These text files are constructed using programming languages, which read as a mix of English keywords and mathematical formulas with various numbers, symbols, and idiosyncratic spacing. For example, below is a piece of code written in Java, built to compute all kinds of three letter combinations ranging from A-Z, AAA, AAB, AAC, etc.²⁷

```
public class Permutation {  
    public static void main(String[] args){  
        char[] permutationArray = new char[3];  
        permute(permutationArray, 0);  
    }  
}
```

²⁷ My thanks to Alexander Kempton for writing this little piece of program for me.

```

static void permute(char[] permutationArray, int x){
    for(int i = 65;i<91;i++){
        permutationArray[x] = (char)i;
        if(x<2){
            permute(permutationArray, x+1);
        }
        else{
            String permutationString = new
String(permutationArray);
            System.out.println(permutationString);
        }
    }
}
}

```

This source code, written as a human-readable file in a higher-level language, to be computable, must be translated into another format if it is to be read by a machine. This usually happens with the help of a ‘compiler’, a software program that turns the original source code into ‘object code’, reading simply as a sequence of 0 and 1s. Alternatively, if the code is written in a language used for web development, such as JavaScript or PHP, the code is not compiled but rather translated ‘on the go’ by an ‘interpreter’.²⁸

Code exists primarily at two layers, the uncompiled (source code) and the compiled (object code) layer. Source code usually refers to the uncompiled, non-executable, human-readable commands to a computer as represented in these plain text files. Code is what instantiates and *describes* software. For this reason, one way to begin to

²⁸ Both compilers and interpreters fulfil the same basic function – they translate source code into machine-readable code. Whereas a compiler translates the entire code into an executable program, the interpreter translates the code line-by-line.

understand software is to engage with its source code in some way. Indeed, as Mackenzie asserts:

[W]hat software does and how it performs, circulates, changes and solidifies cannot be understood apart from its constitution through and through as code. Code even defined in the minimalist technical sense as a “rule for transforming a message from one symbolic form (the source alphabet) into another (the target alphabet)”, cuts across every aspect of what software is and what software does (2006: 2-3).

While there is no denying that understanding the workings of source code is a prerequisite for an understanding of software more generally, how the scholar can and should engage with the actual code is an open question. Opinions diverge on this issue within software studies. There are those who claim that code and the interpretation of code need to be at the heart of the study of software, an approach particularly evident in the notion of *critical code studies* (see especially Marino, 2006; but also Berry, 2011).²⁹ On the other hand, there are those who do not feel that the study of software benefits from the study of source code – an opinion to which I subscribe. However, given the role source code plays in (partly) determining and constituting software, and the salience of the notion of ‘software as code’, a neglect of code in software studies needs to be motivated and justified. I will therefore explicate my view on code, and outline three main difficulties faced by those who argue that code reading should form part of a software approach to social media. I argue that there is a need to make a distinction between the importance of code and the need for code literacy and code

²⁹ The nascent field of critical code studies, mainly focused around a group of US-based electronic literary scholars, has been concerned with debating how to analyse source code using hermeneutic methods. As Mark Marino, one of the founders of this subfield, says: ‘I would like to propose that we no longer speak of the code as a text in metaphorical terms, but that we begin to analyze and explicate code as a text, as a sign system with its own rhetoric, as verbal communication that possesses significance in excess of its functional utility’ (2006). In recent years there have been several organised attempts by Marino and others to debate and discuss the interpretation of code. The first critical code working group was arranged online (through the social networking platform Ning) during winter 2010, and lasted for several weeks. Every week there was a new theme for discussion, spurred by an invited comment or presentation by a leading scholar in the field. During early 2011, the discussions were continued through a HASTAC-hosted discussion forum, which attempted to ‘develop and practice the reading methods and interpretive moves that can be used to read code’ (see <http://hastac.org/forums/hastac-scholars-discussions/critical-code-studies>). Most recently, during February 2012, a second critical code studies working group took place online (still by invitation or membership only), with weekly round-table discussions and examples of code critiques. I’ve participated informally in all these events, mostly as an observer, but have occasionally also participated in the discussions by posting comments.

readings as an appropriate methodology for understanding sociality on social networking sites. I claim that acknowledging the importance of code does not necessarily commit one to a practice of code reading when analysing sociality on the web.

The meaning of code

To say that code can be read as text implies that code has meaning in and of itself. However, there is a danger of projecting too much meaning onto something that arguably is a seemingly static unambiguous sign system. To suggest that code can be read as text seems to suggest that source code is to programming language what text is to natural language. Language, whether programmable or natural, can be seen as a carrier of meaning, but an important difference between the two resides in the fact that programming languages do not have the properties of ambiguity, polysemy, or vagueness, as natural languages do. The implication of this is the following: source code is isomorphic, in that there always is a 1:1 relation between a piece of code and what it signifies in a given language. The following example is written in the open source programming language Processing:

```
void setup() {  
  size(400,400);  
}
```

```
void draw() {  
  background(200);  
  fill(255,200,0);  
  noStroke();  
  ellipse(200,200, 200,200);  
}
```

This code simply means ‘draw an orange circle’, and nothing else. It cannot, in another context, be read as ‘draw a green square’. It does not lend itself to interpretation. The same cannot be said of natural languages, where a statement such as ‘he is here now’ can mean a number of things, depending on the context. Nothing analogous to such *context sensitivity* can be found in the case of programming languages.

Whereas natural languages are sites for the struggle of meaning, programming languages are precise, mathematical, and functional. Natural languages are prone to ambiguity, which leaves texts written in natural language open to interpretation. Reading the sentences of programming languages, by contrast, does not leave room for such interpretation, as a syntax string of code, written in a particular language, can only mean one thing. As Florian Cramer (2008) stresses, programming languages are nothing more than human languages for machine control, characterised by their purely syntactical rather than semantic natures. Computer-control languages therefore do not carry any meaning beyond being a purely cultural-social convention to assign symbols to machine operations. According to this view, software has no semantic denotation. Meaning in natural languages is something that humans acquire with reference to the world, while meaning in programming languages is something derived from, and inscribed by, the programmer.

The isomorphic relationship between code and its meaning aside, there are ways in which the apparent singularity of code can be contested, and code can be seen as ‘meaningful’. For example, one might, reading the way a code is written, infer the identity of the coder and his or her style of programming. Code may also be analysed with reference to the specific software development paradigm that is being used (i.e. structured programming, object-oriented programming)³⁰ and the development methodology used (i.e. agile software development, extreme programming).³¹ Code can also be read for its elegance. As Matthew Fuller points out, the notion that

³⁰ For an account of structured programming see for instance Dahl, O. Dijkstra, E.W. and Hoare, C.A.R. (1972). For more on the object-oriented approach see Crutzen, C., Kotkamp, E. (2008).

³¹ See Anders Løvlie’s PhD dissertation for an interesting overview and appropriation of agile methods into media studies. Løvlie, A. (2011). Also, the agile manifesto detailing some of its principles. <http://agilemanifesto.org>. For an account of extreme programming, see Chapter 7 in Mackenzie, A. (2006). *Cutting Code*.

programs can possess the quality of elegance makes it possible to evaluate software on the grounds of how the problem is approached (2008: 87-92).³² In addition, David Berry offers a comprehensive account of some of the ways in which code can be approached in *The Philosophy of Software* (2011). In response to the question of what exactly to look for in code, which lines should be read, etc., Berry suggests that one may analyse the *commentary sections* of code in a discursive fashion. As Berry points out, ‘these textual areas are used to demonstrate authorship, list collaborators and document changes’, offering a kind of historical record and narrative of the code in question (2011: 54).

While these ways of looking at code do not preserve the aforementioned parallel to literary interpretation, they do indicate that there is potential value in reading code in and of itself. Holding source code to be the ‘proper text’ of software, though, seems to imply further difficulties in regard to causality and signification. As Chun has forcefully argued, the belief in source code as the source or logos of software turns source code into a fetish. Source code only becomes source after the fact, and should therefore be treated more as a *re-source* than as the cause for computation (Chun, 2011a).

Chun appears to claim that there is a tendency among critical software scholars to see source code as something that provides the truth about and essence of software. In many respects, efforts to locate an understanding of software in reading and interpretations of the source code raises more questions than it answers. What exactly is there to interpret in code? Does the analyst need the entire code, or just a few lines? Which lines would be the most relevant or representative and why? What constitutes worthy code? What about the source code of proprietary systems, such as Facebook?

It is no doubt important for new media scholars to understand the concept of code, the ways in which it is constructed, and how it operates in order to be able to appreciate the kind of work that software does, or is capable of doing. I do not believe, however, that reading source code will bring us closer to an understanding of sociality in networked software-mediated spaces, for the reasons outlined above. In addition to

³² Elegance, as first proposed by Donald Knuth in *Literate Programming*, can be measured by four criteria: the leanness of the code; the clarity with which the problem is defined; sparseness of use of resources such as time and processor cycles; and implementation in the most suitable language on the most suitable system for its execution (Fuller, 2008: 87).

the isomorphism of code and its meaning and the methodological problems associated with the selection of a unit of analysis, there is a further problem connected to the *access* to code, especially in regard to proprietary software services and black-boxed code. As Lev Manovich points out:

The attraction of “reading the code” approach for humanities is that it creates an illusion that we have a static and definite text that we can study [...] But this is an illusion, and we have to accept the fundamental variability of the actual “software performance” (2012: 18).

Far from dealing with static text, the study of social networking software is fundamentally the study of something that is inherently dynamic and changing, depending on users’ actions. Manovich elaborates: ‘Especially in the case of large-scale commercial dynamic web sites such as amazon.com, what the user experiences as a single web page may involve continuous interactions between dozens or even hundreds of separate software processes’ (Manovich, 2012: 17).

Taking these insights seriously may lead the scholar in another direction, towards the specific architecture that produces the conditions for these interrelations between code and user. Rather than study source code in order to understand how software signifies meaning in the context of dynamic web sites such as Facebook, we should, then, take a closer look at algorithms.

Algorithm

Computer programs are essentially algorithms, understood as a set of instructions for solving a problem or completing a task. These instructions are not given at random, but follow a carefully-planned sequential order. Algorithms are frequently described as and compared to recipes, or as a step-by-step guide that prescribes how to obtain a certain goal, given specific parameters.³³ Every source code is comprised of algorithms, in the sense that programming languages are languages designed to write algorithms. The Java code cited earlier, for instance, seeks to produce all possible three letter combinations using the letters A-Z. The algorithm, then, is all the various steps given to the computer in order to reach this desired output.

³³ In this sense, algorithm signifies a universal idea. As Jean-Luc Charbert writes in his book *A History of Algorithms*: ‘Everybody today uses algorithms of one sort or another, often unconsciously, when following a recipe, using a knitting patterns, or operating household gadgets’ (1999: 1).

The algorithm is ‘independent of programming languages and independent of the machines that execute the programs’ (Goffey, 2008: 15). The same type of instructions can be written in the languages C, C#, or Python, and still be the same algorithm. This makes the concept of algorithm particularly powerful, given that what an algorithm conceptualises is an assumption inherent in all software designs about order, sequence, and sorting. It is the actual steps that are important, not the wording. In addition, the steps specifying the algorithm are usually just one of many possible solutions to the same problem. For any computational process, the algorithm must be rigorously defined, i.e. specified in such a way that it applies in all possible circumstances.

The algorithmic recipe is not exactly like a recipe for a tomato sauce, though. It is much stricter in the sense that the steps provided are conditional; they must be followed in exactly the way prescribed in order for the program to work. That is, ‘any conditional steps must be systematically dealt with, case-by-case; the criteria for each case must be clear (and computable)’ (Scriptol, 2012). Unlike the tomato sauce recipe, the computational notion of algorithms hinges on the principles of iteration, recurrence, and recursivity.

Some of the common techniques used to express these instructions in a programming language, such as Java, are various control flow statements and the grouping of code into subroutines. In the Java code below, we see some basic examples of these techniques. Here I will quickly guide the reader through the example with a description of what the algorithm does, by providing a ‘commentary code’ indicated by the //.

```
// In programming languages such as Java, code is structured into classes, which constitute blueprints from which objects can be created. In this program, the Permutation class contains all the code.
```

```
public class Permutation {
```

```
// The main method is the first method that is called when the program is run.
```

```
    public static void main(String[] args){
```

```
// Here, an array (a list with space in memory for a specified number of data items of a specified type) is created, in this instance a list with space for three character items.
```

```
        char[] permutationArray = new char[3];
```

```
// Here we start the function call.
```

```

        permute(permutationArray, 0);
    }
static void permute(char[] permutationArray, int x){

// Here we start a for-loop, which repeats instructions a specified number of times.
    for(int i = 65;i<91;i++){

// We start by adding the value of an array into the variable x. The first loop will start
at 0 with the value A(65). We start by adding a value into x of the permutationArray.
In the first call of the function/method/subroutine, x will be 0, and in the first iteration
of the loop the value entered into the array will be A (which has a numerical value of
65).

        permutationArray[x] = (char)i;

// Here we start an if-then-else control flow statement. If x<2 means that the
statements in the if clause will only be executed as long as x is less than 2. We start at
0, and then we then move onto the next number the next time the function is called.
This is the basic concept of a recursion, the process of repeating the same procedure
until a specified state is reached. When we have arrived at x = 2, the recursion stops
with the 'else clause', by not calling the permute function again. The three letter
combinations starting with 'A' will be displayed, and the loop continues with 'B'.

        if(x<2){
            permute(permutationArray, x+1);
        }
        else{
            String permutationString = new
String(permutationArray);
            System.out.println(permutationString);
        }
    }
}
}
}
}

```

Because of its abstract nature, the notion of an algorithm is difficult to grasp. Google maps directions is one example of how to think about an algorithm. Say you want to get from point A) Oxford circus, to point B) Trafalgar Square. You query Google maps in order to get the walking directions. What Google feeds you is a detailed

description of how to get from point A to B. In six steps, it will tell you where to turn right, where to turn left, and how many miles each of the proposed steps will take until you finally arrive at Trafalgar Square. However, the algorithm could potentially guide you through a completely different walking route, as there are almost infinite ways of walking from Oxford Circus to Trafalgar Square. Given that there are a number of different priorities and parameters that could influence which route will be the most relevant for your purpose, the steps contained in the algorithm will necessarily have to take these conditions into consideration when deciding how to take you from A to B. For example, should it be the most time-effective route or the one with less traffic? In the case of Google maps, the right solution, according to the algorithm, is to give you the most time-effective route. But perhaps time-efficiency is not your highest priority; perhaps you would have preferred to take the most scenic route. In this case, an altogether different algorithm would be needed – one that Google does not provide. The point is that while the goal is still getting from A to B, the solution provided depends on the specific circumstances, assumptions, and priorities embedded in the algorithm.

Algorithms, then, are implemented to control the flow of actions. As Wendy Chun points out, an algorithm is a ‘strategy, or a plan of action – based on interactions with unfolding events’ (2011a: 126). Algorithms act in response to inputs of data, which produce an outcome of structured data. I used Google maps to illustrate the basic logic of how algorithms function, but algorithms themselves are at work in almost every web site and platform today. The impact of algorithms can hardly be overstated, as they are used to sort, rank, recommend, suggest, classify, predict, and cluster items, data, things, and people. Functioning as associative devices, algorithms have the power to organise and arrange relationships in important ways. Yet, their presence and influence on the current information ecology is rarely discussed within media studies. In this dissertation, I will demonstrate how the conceptualisation of software at the level of the algorithm offers a particularly fruitful way to examine the programmability of sociality in social networking sites (see Chapter 6 on visibility and the EdgeRank algorithm in particular).

Executable and execution

Software is both text and procedure. Software runs and executes, meaning that code has an *executable existence*. In its executable form, software has the capacity to become visible as part of graphical user interfaces. Source code constitutes one side of software; the other side involves the procedural operation of code. As Wendy Chun points out, however, the action that makes code executable is not a trivial action. The source code and the executable are not mathematically identical, but rather logically equivalent. For example ‘some programs may be executable, but not all compiled code within that program is executed; rather, lines are read in as necessary’ (Chun, 2011a: 24).

Transforming source code into an executable file is not a one-step process. It usually follows multiple steps, which include translation as well as the involvement of other software programs, such as compilers and linkers. The ‘object file’ created by the compiler, is an intermediate form and not directly executable. In order for a program to be executed, another device called a linker must combine several object files into a functional program, or .exe files.³⁴ Whilst the technicalities are not the main issue here, what is important to point out are the different levels of existence at play when talking about software.

Software is quite literally the gathering or assembling of different code files into a single executable. While software might appear to be a single entity, it is fundamentally layered, and dependant on a myriad of different relations and devices in order to function. Its structure can be compared to that of an onion; a computer system comprises ‘many distinct layers of software over a hardware core’ (Ceruzzi, 2003: 80). It consists of an ‘[a]pplication on top of operating systems, on top of device drivers, and so on all the way down to voltage charges in transistors’ (Chun, 2011a: 3).

Conceptualised as execution, software *does* something. It performs the encoded instructions, thereby making things happen. This apparent ability to make things happen has been one of the focal points for scholarly discussion of software within the

³⁴ In a program written in the programming language C, the different steps can usefully be illustrated by the program’s file-naming convention. The source code file ends in ‘.c’, the object code ends in ‘.obj’, and the executable files end in ‘.exe’.

humanities. In particular, these debates have conceptualised software as language, seeing its force in terms of the *performative* and *performativity* of code (see Galloway, 2006; Hayles, 2005, Mackenzie, 2005; Mackenzie and Vurdubakis, 2011).

Derived from J. L. Austin's account of language, the concept of the performative refers to the notion that language and speech not only express things about the world, but also have the power to act upon that world (Austin, 1962). In Austin's account, certain speech acts (illocutionary acts) have a performative function, meaning that they do what they say.³⁵ In other words, performatives are statements that instigate their utterance. To claim that code is performative, then, implies a functional analogy between code and natural language. In a much-cited passage, Galloway argues that 'code is a language, but a very special kind of language. Code is the only language that is executable [...] it is the first language that actually does what it says' (2004: 165-166). In a similar vein, N. Katherine Hayles argues that code has a strong illocutionary quality:

Code has become arguably as important as natural language because it causes things to happen, which requires that it be executed as commands the machine can run. Code that runs on a machine is performative in a much stronger sense than that attributed to language.

This performative notion of code posits the meaning of software in terms of its function or executability – at the level of the machine. As Galloway puts it, 'code essentially has no other reason for being than instructing some machine in how to act' (2006: 326).

While debate about the extent of which software may usefully be compared to language is theoretically and conceptually interesting, it does not bring us much closer to understanding how software can be said to do work in the world – beyond causing changes in machine behaviour. The usefulness of the performative legacy lies not in terms of code's effects on the machine, but rather in the ways in which the concept has been elaborated upon and reconfigured into the notion of *performativity*. Whereas the strictly linguistic notion of performative assumes a distinction between language and force, the concept of performativity does not. Performativity, as it has been

³⁵ This is to say that propositions may constitute the objects to which they refer. For example, in saying 'I promise', the promise is enacted by means of uttering the words; it is saying and doing at the same time. Likewise when the pastor declares 'I pronounce you husband and wife', the act is performed by declaring the words.

elaborated on in much critical and cultural theory during the past two decades or so (see Butler, 1990; Callon, 1998; Pickering, 1995), designates a way to depict the world not as an already existing state of affairs, but rather as a *doing* – ‘an incessant and repeated action of some sort’ (Butler, 1990: 112).

As Mackenzie (2005) has argued, the conduct and practices of code work in a similar way. By performing its encoded instructions, software enacts the acts that it represents. As such, software hinges on the kinds of ‘looping effects’ that Ian Hacking (1995) talks about: positive feedback effects that sustain the entities they postulate.³⁶ The performativity of software not just pertains to the linguistic relation between different layers of code as ‘doing what it says’, but more importantly, to the ways in which software contributes to framing the world in certain ways.

This way of conceptualising software as execution provides a productive way to study software as an active participant engaged in ‘ways of worldmaking’ (Goodman, 1995). In the chapters to come I will address the ways in which software produces the conditions it merely perpetuates to describe, and how it becomes the condition and occasion for further action (see Chapter 5 on attention, 6 on visibility, and 7 on friendship). As an operational and performative object, software, as Mackenzie points out, ‘coordinates the encounters between specific social actions pertaining to information and communication networks’ (2006: 76), something that will be further explored in Chapter 7, and Chapter 8 on APIs.

Practice

Just as, for example, literature is not only what is written, but all cultural practices it involves—such as oral narration and tradition, poetic performance, cultural politics - software is both material and practice (Cramer, 2005: 122)

In this final section, I draw attention to the ways in which software needs to be understood as *practice*. As I have already suggested, the executable level of software points towards one such conceptualisation of software as practice, in the sense that multiple processes and actors constitute software into an operational form. Here, I

³⁶ While Hacking talks about ‘human kinds’, or the social construction of the social, through the ways in which new ways of classifying people also changes how people can think about themselves, the self-perpetuating logic is also evident in the ways in which software performs.

take practice to refer to the various ways of *doing* that software implies, including the *production* and *use* of software. As Adrian Mackenzie claims, ‘software is a neighborhood of relations whose contours trace contemporary production, communication and consumption (2006: 169). Not only is software produced through maintenance, problem solving, and upgrading. Software, then, is seldom produced once and for all. In fact, if we use the word ‘program’ as von Neumann and his colleagues used it, to describe the ways in which software constitutes a practice become even more apparent. A program, then, is not so much a bounded object, as it is an activity, the developing result of continuous assembling and organising.

The production of software involves a plethora of actors and the interests and struggles between these. Software always implies an originator, whether or not ‘the originator is a programmer, webmaster, corporation, software engineer, team, hacker or scripter, and regardless of whether the originator’s existence can be forgotten, sanctified or criminalized’ (Mackenzie 2006: 14). While the notion of the solitary programmer and hacker is still a common way of framing software as practice, in most cases, the production and practice of software involves a process of collective manufacture (Kichin and Dodge, 2011). We can see these two sides of software production strongly represented in the way the history of software is written. Two types of myths are particularly salient: on the one hand, the myth of individual programmers and the narratives about collective programming methods, paradigms, and organisation on the other. Software often involves many different layers of production, phases, places, management, and organisation. Perhaps the most-discussed and extreme version of software as a collective manufacture can be found in the case of *Free, Libre and Open Source software* movements (FLOSS).

Importantly, software rarely constitutes a one-off event or finished product, but rather needs to be seen as an ongoing effort, a process of becoming. As such, software can be considered a *project*. As projects, software requires care, maintenance, updates, revision, and work. The continuously evolving, developing, and transforming landscape of software poses a challenge not only to the producers and users of software, but also to the researcher wishing to study it (something I will touch upon in the next chapter). For programmers, the challenge consists of trying to keep up with the latest developments, including new programming languages, code libraries, and development environments (Kitchin and Dodge, 2011: 35; Mackenzie, 2006).

It is important to point out that software as practice refers not merely to the production of software by a programmer, but also to end-user practices. Users play an important and much-debated role in software development – from the beginning of product development, to the testing of the end product (see Halonen, 2007). But users also play a much more immediate role and participate in the ongoing development of software. In the case of algorithmically-driven sites such as Facebook, user practices are crucial to the development and maintenance of the underlying coded systems, as they constantly feed the system with new data. As Mukund Narasimhan, software engineer of Facebook, tellingly suggests: ‘Everything in Facebook is a work in progress’. The models Facebook uses to design the system are evolving because the data is changing. This means that the exact ways in which the algorithms work are also constantly tweaked, because of the fact that everything else changes (Facebook, 2011a). For end users, one might say that this mutability of software constitutes an ethics of constant care. The many changes of the Facebook platform carry with them a responsibility on behalf of the user to stay up-to-date and make the necessary adjustments suggested by the software.³⁷

Software as practice also refers to what might be termed a *phenomenology of software*, in terms of how software is a matter of lived experience and makes a difference to people’s everyday lives. Software functions in a traversal way; it cuts across the boundaries of what is commonly taken to be its formal existence. As practice, software needs to be understood through its fundamental interconnectedness to everyday life, rather than as an object that is detached from it. While the conceptualisation of software as practice cross-cuts the discussions to come in a mostly implicit manner, Chapter 8 explicitly addresses the ways in which software constitutes a conflictual field of lived experience.

³⁷ In Chapter 5, I will discuss some of the most salient changes to the Facebook platform over the past years. It is not the changes in and of themselves that will be of interest in that chapter, but rather the ways in which the protocological infrastructure organises attention. However, through the work of documenting the amount and nature of the changes to the Facebook platform, it can be argued that changes in the software affect practices in fundamental ways. For example, one of the recent platform changes called the ‘Timeline’ should not only be thought of as a complete redesign of the personal profile feature, but also as a change that completely transforms the conditions for self-presentation.

Concluding remarks

In this chapter, I have argued that software needs to be seen as a shifting nexus of relations that comprises certain prevailing material and cultural characteristics. Simply conceptualising software as a set of instructions that tell the machine what to do fails to take into account the variable ontology of the thing that we understand by software. Software should not be seen as mere code, nor can it simply be reduced to algorithms or executable output. Understanding software implies an understanding of the interaction and contestations of numerous actors, both human and nonhuman. Thus, it is important to understand not only how software embodies different relations, but also how it has the capacity to be productive and generate further relations (Fuller, 2003: 63). This “expanded” notion of software ultimately points to the ways in which software can fruitfully be conceptualised and analysed by considering not only its material existence, but also through the various ‘forms of contestation, feeling, identification, intensity, contextualizations and decontextualizations, signification, power relations, imaginings and embodiments that comprise any cultural object’ (Mackenzie, 2006: 5).

The purpose of this account of the nature of software has been to introduce a way to understand software that can serve as a framework for the chapters to come. In addition, I have been motivated by a desire to provide a fuller, more explicit understanding of the key theoretical notion behind software studies: *software* itself. The complexities revealed by a close analysis of the concept of software, results in there being a number of methodological challenges scholars will encounter when studying software from a humanities perspective. In the next chapter, I grapple with some of these, outline how they may be dealt with, and introduce the methodological framework and methods I apply in my analysis of Facebook and Twitter.

Chapter 4. Methodological framework

In this chapter I introduce my methodological approach to the study of how software fosters sociality on social networking sites. I call this approach ‘technography’. Broadly speaking, I approach the complex ways in which software organises, shapes, and assembles sociality in the context of social networking sites in a way that is similar to anthropological methods of ethnography. This is to say that technography, as I use the term here, is a way of describing and observing technology in order to examine the interplay between a diverse set of actors (both human and nonhuman). In contrast to ethnography, with its focus on ‘documenting people’s beliefs and practices from the people’s own perspectives’ (Riemer, 2009: 205), technography has technology as its perspective; specifically, the norms and values that have been delegated to and materialised in technology. Here it is important to point out that technography should not be thought of as method distinct from ethnography, but rather as a mode of inquiry into the meanings embedded in the mechanics of technology that *make use of ethnographic methods*. The main difference between ethnography and technography lies in the researcher’s perspective. While the ethnographer seeks to understand culture primarily through the meanings attached to the world by people, the technographic inquiry starts by asking what the software itself is suggestive of.

While technography constitutes the main approach of chapters 5, 6, and 7, which focus on the programmed sociality of Facebook, I also rely on more ‘traditional’ media ethnography in Chapter 8, which deals with the Twitter APIs. In order to create an understanding of the ways in which power articulates through software, I believe a combination of perspectives, methods, and sites of analysis provides fertile ground for a diagrammatics of software in social networking sites. Thus, my methods are not confined to a structural analysis of Facebook mechanics, but importantly include the perspectives of human actors who work with software on a daily basis. The following chapters form responses rather than definite answers to my research questions. I follow Matthew Fuller’s assertion in his introduction to the software studies lexicon: ‘The purpose [...] is not to stage some revelation of a supposed hidden technical truth of software [...] but to see what it is, what it does and what it can be coupled with’ (2008: 5).

Empirically and analytically I follow a grounded approach, where the theoretical reflections are developed from the data collected and the field of study.³⁸ In my descriptions of the workings of software processes and mechanisms embedded in social networking sites, my analytical focus is on how the operations make a difference to the ways in which collective associations form and unfold in the context of these software-mediated spaces. The data used in this dissertation is drawn from various empirical sources, which include auto-ethnographic observations of the sites, online interviews with software developers, aggregated data concerning the software mechanics of the sites, and analysis of relevant text sources concerning the sites (i.e. blogs, technical specifications, mailing lists, developer talk, manuals etc.).

In this chapter, I will first outline my understanding of technography as a methodological framework useful for the analysis of the micropolitics of power through software. In the second part of the chapter, I provide an account of the various methods used, including the reading and analysis of the operational logic of software and online interviews.

Technography: A descriptive-interpretative approach

How to make sense of the convoluted nature of software, of the ways in which it signifies and produces the conditions for the intelligible and sensible? As with human cultures, there is no single answer or way of going about researching the power of coded systems. Just as there is a need for media and communication scholars to become more attentive to the ways in which software makes a difference in mediation and communication today, there is a need to expand the discussions regarding the most appropriate methods and means for researching the subject-software continuum. Here, I propose one possible research strategy – technography - that fuses well-known

³⁸ There are different kinds of grounded research approaches; the basic principles are however the same – the gradual accumulation of new data/readings that builds on the coding and interpretations from the foregrounding round of data collection/readings. The Grounded Theory methodology (Glaser and Strauss, 1968), for example, is a common strategy used within the social sciences, whereby the research process is guided by the constant comparison of data collected in order to ensure well-founded coding categories. See also Charmaz (2006) for a good overview on Grounded Theory. In a similar vein, hermeneutic approaches often used within the humanities also hinge on processes of iterative interpretations, where ‘readers gradually work out the categories of understanding in order to arrive at a coherent interpretation’ (Bruhn Jensen, 2012: 29).

methods from the humanities and social sciences to make sense of ways in which software shapes social life online.³⁹ I see technography as a descriptive-interpretative approach to the understanding of software, rooted in a critical reading of the mechanisms and operational logic of technology.

If ethnography can be understood as a way of understanding cultural practices in the context of everyday life, technography adds technology to the equation by making it a prefix. We would thus say that technography is a way of understanding *techno*-cultural practices in the context of everyday life. Technography makes use of ethnographic methods, so why not just say ‘ethnography of technology’, as other authors have done (see Kien, 2008)? There are two reasons for this. First, I want to refrain from using ‘ethno’ at all, as what I am interested in is not so much the people’s own belief systems or lifeworlds, but rather the ‘lifeworld’ of technology. In other words, I am interested in what can be learned from the connective and associational logics of algorithms and protocols themselves. I ask what the software can be said to be suggestive of, and which underlying assumptions, norms, and values are embedded in the technologies used in everyday life. Therefore, a second reason for dropping the reference to ethnography is to emphasise the importance of engaging in descriptive-interpretative studies of technology, without necessarily having to include particular users’ voices or belief systems. Just as technology becomes meaningful when users are studied for their practices involving technology, the user is always already implied when studying technical mechanisms and functions.

My understanding of technography is indebted to methodologies such as Actor-Network Theory, virtual ethnography (Hine, 2000; 2011), the ethnography of infrastructure (Leigh Star, 1999) and multi-sited ethnography (Marcus, 1995). My main motivation for using the term technography rather than ‘techno-ethnography’ or ‘ethnography of technology’ is to point to the structural, material, and architectural

³⁹ While the term technography has been used by others, my understanding of the term differs from the two or three occurrences of the term that I was able to come across in a literature search. Technography has previously been described as an ‘ethnography of technology’ (Kien, 2008), as a research strategy aimed at uncovering the constructed nature of technoculture (Vannini and Vannini, 2008), and as a way to explore artefacts in use (Jansen and Vellema, 2010). My understanding comes closest to that of Vannini and Vannini, who describe it in terms of a more general attitude towards the attempt to understand the structural aspects of complex technocultural layers. However, as I will detail in this chapter, I use technography in quite a literal sense, making the descriptions of technology the basis for an interpretation of how software organises sociality on social networking sites.

aspects of media and software as opposed to the ethnographic spirit of inquiry so deeply entrenched with understanding cultural practices (see Hine, 2011). Rather than seeking to describe and observe people, as the etymology of ‘ethno’ implies, technography provides a more accurate description of what I set out to do: namely, to observe, describe, and interpret software on its own material-discursive terms.

From Actor-Network Theory and ethnography I borrow the emphasis on description. One of the key research strategies of ANT is to provide detailed descriptions of the intermingling of human and nonhumans constitutes, where specific sociotechnical events are explored for the plethora of actors involved (Latour, 2005: 128). Viewing software as an event shifts attention from what it is to what it does, to the difference it makes between a before and an after by its capacity to question something, render something problematic, or makes it possible to feel, perceive, act, or know differently (Mackenzie, 2005: 388).

In this dissertation, I utilise a number of data-collection techniques that are typical of ethnographic research, including observation, writing ‘field notes’, and interviewing and reading available reports and records. As Rimes explains it, ethnography is always eclectic in its employment of multiple methods and techniques, as what counts as relevant information is not given in advance (2008: 209). Similarly, what counts as relevant sites for data collection with regards to describing the relationships between software and subjects on social networking sites is not a given. Software exists on many levels, meaning that it cannot be easily tracked or accessed. An alternative to researching bounded sites or objects is to ‘follow the thing’ (Marcus, 1995) or the ‘actors’ and ‘associations’, as Latour suggests (2005).

Thus, I have examined software as it exists on many different levels, not only as part of a more or less concrete thing, but also as part of the many associations and sites through which it appears. In this dissertation, I have found software at the level of algorithms, protocols, APIs, programmers, blogs, media reports, technical specifications, recorded engineering talks, conference papers, developer discourse, mailing lists, buttons, user interfaces, information feeds, features, and functionalities.

Marcus developed the notion of ‘multi-sited ethnography’ as an exercise in mapping terrain that moves across many different sites, as opposed to the ‘traditional single-site mise-en-scene of ethnographic research’ (1995: 99). Influenced by postmodern

thinking such as Foucault's power/knowledge and Deleuze and Guattari's rhizome, a multi-sited ethnography in Marcus' terms takes as its starting point that the social is never fixed, but rather is the product of temporarily stabilised connections between heterogeneous elements. In this sense, any attempt at understanding how power articulates through software must take the distributed and multi-situated nature of software into account. The diagrammatics, or cartography of power, is thus designed around the various 'chains, paths, thread and conjunctions' in which the objects of study form a presence (Marcus, 1995: 105).

I adopt the view that technical objects and elements afford meanings and possibilities for action, based on how they work and function. My methods are therefore influenced by approaches to media that explicate the need to read and interpret technology based on its materiality and procedurality. Here, I refer to Galloway's (2004) close reading of Internet protocols, Ian Bogost's (2007) analysis of the 'procedural rhetoric' of computer games, Noah Wardrip-Fruin's (2009) notion of 'operational logic', and N. Katherine Hayles' (2004) media-specific analysis. These approaches suggest a way of attending to the materiality of media by looking at the ways in which the techniques and procedures both constrain and enable certain forms of being together and forming attachments. For example, much in the way Galloway 'attempts to read the never-ending stream of computer code as one reads any text, decoding its structure of control as one would a film or novel' (2004: 20), I have attempted to look at the software specificities of Facebook and Twitter in order to explore how sociality is governed and organised on the platforms.

This is similar to the strategy proposed by Latour, where technological mechanisms should be read directly in terms of deciphering its inscriptions (see Mackenzie, 2002: 211). As Mackenzie exemplifies:

Such a reading is possible because various traces and semiotic systems cluster around technical artefacts and ensembles. Different codes, protocols and conventions compose much of the infrastructure involved in computation, for instance. These layers of inscription mark the diverse negotiations which a technical object embodies (ibid.).

Inspired by notions of specificity and technical inscriptions, the case studies in the following chapters are organised around specific software objects and the social ties that are built through them (see Latour, 2005: 80). I look at how a particular technology functions in order to get at the ways in which technical mechanisms

manage, modulate, and distribute possibilities for action and interaction on social networking sites. Seeing software as an event connects in various ways to the theoretical framework sketched in Chapter 2. Ultimately, I believe, software can be mapped by examining the ways in which it creates new forms and constellations ‘that catalyze previously existing actors, things, temporalities, or spatialities into a new mode of existence, a new assemblage, one that makes things work in a different manner and produces and instantiates new capacities’ (Rabinow 1999: 180).

Letting the software ‘speak’

Technography as it is developed in this dissertation focuses on letting the software actors ‘speak’ by following the ethnographic principle of description. I suggest that software speaks by means of what it is capable of doing, its technical specificities, and the ways in which it forges relationships. My use of the verb ‘to speak’ should not be taken as an attempt to anthropomorphise software, but rather as a way of taking into account the descriptions offered by technologies. As Bruno Latour puts it, ‘specific tricks have to be invented to make them [technology] talk, that is, to offer descriptions of themselves, to produce scripts of what they are making others –humans or non-humans – do (2005: 79).

What kinds of descriptions does software offer? How does software speak? On a basic level, software itself is a description. Software/code is the only language that does what it says. Galloway states that software is self-referential. In this sense, no specific tricks have to be applied to make software ‘speak’, as its production already exists as an output to be studied on the level of the graphical user interface. The executable layer of software translates into visual outputs on our computer screens, which constitutes an important site for the study of software.

When embarking on this research, I realised how quickly these sites change, how often the layout and interface design changes, how many new features and functionalities are added, and how quickly some of them disappear again. The technical field implied by a technography is just as ‘messy’ as the cultural field that is of interest to the ethnographer. This means that technical objects and elements are not static things, but rather are characterised by mutability and conflicting views and practices.

Confronted with ever-changing and mutable platforms, the need for documentation or taking field notes becomes crucial. To this end, I have regularly taken screenshots of my Facebook homepage, the news feed in particular, and documented new features, functions, and changes. These fragmentary snapshots of Facebook proved to be a useful way to document the mobile and dynamic nature of the site. This method of documentation – providing detailed descriptions of the field with systemised screenshots and note-taking – has much in common with the taking of ethnographic field notes, and has proved to be a valuable way to track and map the ontogenetic nature of the medium and software of these social networking sites.

On another level, information about what software is capable of doing exists as part of technical specifications that detail certain functionalities and design choices. How and what specific software, algorithms, or protocols are supposed to do is often described as part of technical specifications, engineering papers, recorded industry talks, media reports, patent applications, blog posts, etc. As such, these types of software descriptions form crucial sites for a technographic approach to data collection in the context of this dissertation.

The many social media industry blogs dedicated to chronicling the lives of sites such as Facebook and Twitter have proved to be particularly important sites for collecting data relevant to an understanding of the structural aspects and mechanics of social networking sites. As Alice Marwick (2010) points out in her dissertation on the San Francisco technology scene, social media industry blogs form an important part of the cultural formation of ‘social’ software, as they document the important interactions and events that take place around powerful tech companies. Blogs, including Mashable.com, Techcrunch.com, and Readwriteweb.com, thus constitute important elements in the ‘neighbourhood of relations’ that Mackenzie suggests describe the complex and circulatory existence of software in general (2006: 171). Therefore, these blogs are not only useful sources of secondary data, but rather must be seen as part of the software assemblage itself. Thus, one important tracking strategy used in this dissertation is to follow news about Facebook and Twitter by regularly reading said blogs.

In addition to blog posts about Facebook and Twitter’s ‘lives’ as software, the different studies draw on diverse materials of ‘software descriptions’ that have informed my understanding of the technical mechanisms behind Facebook and

Twitter. These include technical specifications, policy documents, engineering and developer talks where the specific logics of systems are discussed and presented, protocols and standards, API documentations on Facebook and Twitter developer and engineering sites, general technical information about Internet infrastructure from Request for Comments (RFC) as well as the World Wide Web consortium (W3C), mailing lists, and online forums.

Black box – White box

Letting the software speak in order to read it in the way ‘one reads any text’ (Galloway, 2004: 20) is not without its problems. One of the main obstacles in any attempt to read the underlying software principles of social networking sites pertains to the seemingly ‘black-boxed’ nature of these systems. The *black box* is an opaque technical device of which only the inputs and outputs are known. It is an entity whose functioning cannot be known, at least not by observation, as the blackness of the box obscures vision, challenging any preconceived ideas about what can and cannot be known.⁴⁰ Because the software that underlies some of the biggest media companies today, including Google and Facebook, is what constitutes the assets and values of these companies, the logic of how they are connected naturally remains a highly-guarded trade secret. Today, algorithms are the equivalent of the secret ingredients of Coca-Cola.

My methodological approach to the problem of the black-boxed nature of my objects

⁴⁰ The concept of the black box has a very interesting history, stemming from an atmosphere of extreme secrecy, as part of the military and war machinery of WWII. Historically, the box refers quite literally to the entity containing hidden radar equipment. Electrical engineers confronted with a black box had to deduce its workings and functionality from an observation of the relationship between input and output. In tracing the genealogy of the black box, von Hilgers (2009) describes how the black box initially referred to a ‘black’ box that had been sent from Britain to the USA as part of the so-called Tizard Mission, which sought technical assistance for the development of new technologies for the war effort. This literally black box that was sent to the radiation lab at MIT contained another black box, the magnetron. During wartime, crucial technologies had to be made opaque in case they fell into enemy hands. According to Peter Galison, this culture of secrecy, or ‘radar philosophy’, became a vital engineering technique and also played a key role in the emergence of cybernetics. Galison (1994) describes how the notion of the black box offered Norbert Wiener and his colleagues a way to conceptualise and design complex man-machine systems (i.e. the AA predictor). As we can see, the concept of the black box is inherently linked to secrecy, interpretation, and prediction, as it during wartime became crucial to be able not only to interpret the enemy’s messages and technologies, but also to predict the enemy’s next move.

of study has been to compile data aggregated from auto-ethnographic observations of the interface, with the available information about the mechanics of the sites. I suggest that combining data from the two levels at which software makes itself known, identified above as its execution on the interface and the descriptions of its workings contained in external textual sources, provides a useful way of opening up the black box. This is closely linked to what Ian Bogost calls a ‘white box’ analysis, where one uncovers the computational procedural expressions crafted through code. When neither code nor the exact codified rules of operations are available for direct analysis, methods of accessing the operational principles of systems without extrapolating code need to be developed. Bogost draws on the distinction between what in actual software development is called back-box analysis and white-box analysis to explain his approach:

To watch a program’s effects and extrapolate potential approaches or problems (in the case of testing) in its code is called black-box analysis. Such analysis makes assumptions about the actual operation of the software system, assumptions that may or may not be true. To watch a program’s effects and identify actual approaches or problems in its code is called white-box analysis (or sometimes, glass-box analysis). Such analysis observes the effects of the system with a partial or complete knowledge of the underlying code that produces those effects. Some white-box analysis can be performed without direct access to code. Examples include architectural descriptions from conference presentations about development techniques [...] Publicly documented hardware and software specifications, software development kits, and decompiled videogame ROMs all offer possible ways of studying the software itself (2007: 62-63).

My approach resembles this notion of white-box analysis, in that my reading of software is based on descriptions and observations of the system at the level of the interface, complemented with knowledge of the architecture and underlying functionalities, obtained from available external documents. For example, in the next chapter (5) on attention, I deal with questions concerning the power of Facebook’s Open Graph protocol, primarily by extrapolating from various empirical sources that describe the workings of the system. Drawing on a range of materials that describes how the Facebook platform functions – its underlying design principles and the components of the underlying infrastructure – provides the grounds for analysing

what the software can be said to be suggestive of, and how it organises and distributes the sensible and intelligible.⁴¹

Chapters 6 and 7, which deal respectively with visibility and friendship on Facebook, hinge on a similar approach to the analysis of the functional and operational aspects of the platform. Assembling a plethora of empirical textual sources, and detailing and describing the systems studied allows for an understanding of software based on its properties, mechanisms, and operations.⁴² Akin to ‘media specific analysis’, the Facebook platform is analysed for the kinds of prescribed norms, values, and practice that are grounded in its properties and affordances, and the possibilities for actions provided by the system.

Analytically, I draw on Ian Bogost’s reading of video games as systems of ‘procedural rhetoric’ and Noah Wardrip-Fruin’s concept of ‘operational logics’. Methodologically, both Bogost and Wardrip-Fruin argue for an approach to digital

⁴¹ Here I use technical descriptions from the following sources: The Open Graph protocol specification (ogp.me); Facebook Developer blog (developers.facebook.com); Facebook Engineering (www.facebook.com/Engineering); Conference presentations and recorded talks that explain and describe the architecture of Facebook, as retrieved from various databases, including www.infoq.com, www.ieee.org, and dl.acm.org; Facebook Tech Talks video archive, especially ‘Extending the graph tech talk’ (see Facebook, 2011a) and ‘Hacking the Graph Tech Talk’ (see Facebook, 2010); blogs, including: Techcrunch.com, mashable.com, readwriteweb.com, arstechnica.com, insidefacebook.com, and allfacebook.com; online forums, including Stackoverflow and Slashdot; and the Facebook group dedicated to debating issues concerning the Open Graph. I am a member of this group, and have followed the discussions on a regular basis.

⁴² In Chapter 6 I focus on documents that describe the Facebook News Feed and the EdgeRank algorithm. Here I draw on many of the same sources described in the previous footnote, (the same blogs, forums, and databases). In addition, I draw on the video broadcasts of the technical sessions of the 2010 Facebook developers conference, especially the session called ‘Focus on the feed’, in which Facebook engineer Ari Steinberg talks about the details of EdgeRank. Moreover, social media marketing sites provide helpful sources with regard to technical details, as they often must convince new customers of the need to buy their services based on some new technical reality. Companies such as the EdgeRank Checker (edgerankchecker.com) have sprung up to boost users’ visibility on Facebook’s news feeds. These types of company websites thus offer a source of additional information about some of the technical workings.

In Chapter 7, I rely on the same kind of source materials as used in chapters 5 and 6 (i.e. blogs, conference papers, tech talks, databases). Thus, Chapter 7 repurposes the technical information about the Open Graph protocol explored in Chapter 5, and EdgeRank in Chapter 6. In addition, the chapter looks at the workings and design principles of people-finding algorithms and the ‘Friend of a friend’ (FOAF) approach. As this is a well-known approach within computer science, conference papers describing FOAF in the context of social networking provide a particularly important source.

media that does not hinge solely on digging underneath the screen to address the programmability and expressivity of systems, but rather focuses on the ways in which the software operates. The notion of ‘operational logics’ refers to the behavioural aspects of digital systems, to the ways in which digital media are capable of ‘expressing a position through their shapes and workings’ (Wardrip-Fruin, 2009: 4).⁴³ In a similar vein, Bogost’s notions of ‘procedural rhetorics’ emphasises the need to understand the expressions of computational systems, in order to uncover how the systems form and author certain arguments. According to Bogost, procedural rhetoric is the art of persuasion through rule-based representations and interactions, tied to the core affordances of the computer (2007: ix).

Reverse engineering

In Chapter 6, I combine my descriptive-interpretative technographic approach with a more experimental mode of ‘reverse engineering’. According to Eldad Eilam, ‘[r]everse engineering is the process of extracting the knowledge or design blueprints from anything man-made’ (2005: 3). In that sense, even my overall technographic approach could be considered reverse engineering, in that it aims to identify how a thing works in order to extrapolate its signifying effects. As Eilam further elaborates:

Reverse engineering is usually conducted to obtain missing knowledge, ideas and design philosophy when such information is unavailable [...] Traditionally, reverse engineering has been about taking shrink-wrapped products and physically dissecting them to uncover the secrets of their design [...] In many industries, reverse engineering involves examining the product under a microscope or taking it apart and figuring out what each piece does (2005: 3-4).

While the concept of reverse engineering refers for the most part to actual methods applied in the field of software engineering, I use the term in a more metaphorical sense. Whereas software engineers use methods of code breaking, disassembly, and

⁴³ In his doctoral dissertation, where he originally developed the concept of operational logics, Wardrip-Fruin (2006) points out that we do not have to know how to design software (or even how to read code) to pay closer attention to software and how it works. What is needed is a consideration of the rhetorical nature of the software processes that generate the text that becomes accessible on our screens. The question for Wardrip-Fruin is how to understand digital systems by focusing on process. Reading code will not do, he asserts, as it would merely reproduce a kind of media studies based on fixed objects (Wardrip-Fruin, 2006: 6). Instead, an understanding of processual media requires taking various layers of the systems into account, and interrogating how they operate and interoperate.

decompilation – to name some of the techniques of reverse engineering – my approach remains on the interface level and has nothing to do with software development per se. As I am interested in exploring the power of software in the organisation of collective life in social networking sites, I decided to look more closely at what I take to be one of the most important sites where sociality currently plays out, namely Facebook’s News Feed.

At the time of my research, the News Feed was divided into two separate feeds, the ‘Top News’ feed and the ‘Most Recent’ feed.⁴⁴ Melody Quintana, a specialist on Facebook’s user operations team, described the difference between the two feeds in a Facebook blog post thus: ‘Top News shows popular stories from your favorite friends while the Most Recent shows you all updates from as many as 250 friends’ (2010). While Top News is based on an algorithm that aggregates the content users will supposedly find most interesting and displays the posts as a sorted feed where the apparently most relevant stories come first, the purpose of the Most Recent feed is to display all posts that happen in near real-time.

In order to explore the operational logic of the algorithmically-driven News Feed, I conducted an experiment of reverse engineering, based on a structured comparison of the contents of the two feeds during separate time periods. The fact that there were two separate feeds provided a unique opportunity to explore their workings in relation to each other. Given that the Most Recent feed displayed all updates by friends, the contents of the two feeds could be examined to explore which posts were considered important enough by the algorithm to be displayed on the Top News feed. During April 2011, I began to use screen shots as a method of inquiry to record the ever-changing platform and feeds. Several times a week during April, June, and September 2011, I took screen shots of my entire Top News feed, counted the amount of posts, and compared it to the amount of posts published on the Most Recent feed during the same time period. While I did not take screen shots of the Most Recent feed, as it was simply too time consuming, for every Top News analysis I went to the Most Recent feed to compare and contrast.

For example, I made lists using a computer spreadsheet, where I tried to identify the ‘ranking’ and location of the same post in both feeds. I would compile a list of the top

⁴⁴ As of September 20, 2011, the two feeds were collapsed into a single feed.

20 posts published on my Top News feed at a given point in time, and look for the same posts and their place in the Most Recent feed. My coding scheme for the analysis was based for the most part on knowledge of the workings of the EdgeRank algorithm that govern the Top News feed, which I had been gathering successively over the course of one year prior to the experiment. During Facebook's annual developer conference *f8* in April 2010, Facebook revealed important technical details about the design principles of the EdgeRank algorithm for the first time, outlining some of the main factors used. Based on this information and other documents that reported on the mechanisms and operations of the feeds, I compared the content of the feeds in terms of time decay, nature of the post (what kind of post and how many Comments and Likes the post had generated), and who had posted it (whether a friend or company Page).

This experiment of reverse engineering offered a useful approach to the seemingly black-boxed nature of the Facebook system, and revealed Facebook as more of a 'grey box' than a total 'black box'. Inspired by the early cybernetician W. Ross Ashby, the task is not necessarily to know exactly what is inside the box, but to ask instead which properties can actually be discovered and which remain undiscoverable (see Ashby, 1999). While the exact workings of the algorithm and the platform remain unknown, many useful details are still available. I believe that assembling and bringing together the information extracted from the interface and the text documents offers a useful methodological approach to understand the workings of software. Ultimately, I see the experimental mode of inquiry as a 'trick' in the sense of Latour, in terms of making the software 'speak', or offer a description of itself. Finding ways of making technology reveal itself – letting it speak – constitutes a way to engage in a reading that takes as its starting point the kinds of things that technology itself can be said to signify and suggest.

One of the difficulties, however, with trying to extract the design principles and underlying logic of how the system works, is the uncertainty associated with the mutability and ontogenetic nature of the software. As the algorithms of Facebook are fundamentally individualised and personalised, auto-ethnographic observation of the interface is confined to the 'me-centric' view of the researcher's own account and

network.⁴⁵ My observations and attempts at reverse engineering the technical blueprints are always already skewed towards the subjective view of the researcher. Facebook adds a whole new dimension to the methodological challenges typically faced by ethnographers in terms of attempting to understand culture from the insider's view. In the case of Facebook, the *medium fundamentally follows the researcher* and not merely the other way around. Therefore, one's view is always already obscured by the fact that Facebook, by virtue of its algorithmic logic, functions as a shadow. In the age of 'algorithmic culture', seeing is not just filtered through our own ideas as Rimes (2009) suggests is the case in typical ethnographic research, but is also filtered through the prescribed logic of the systems that we seek out to study.

Interviewing developers

In order to gain insight into the multiple ways in which power articulates through software, approaching software at a micro-level of lived experience was central. As one of the premises of software studies is to understand software at the multiple levels of existence, I decided to approach people who arguably 'know' software – developers and programmers. For the purpose of assembling data about the ways in which power articulates through software in social networking sites, I conducted 20 e-mail interviews with web developers who use the Twitter application programming interfaces (see Appendix 1). I had two main reasons for conducting interviews with software developers. First, as there is an overwhelming focus on end-user research in social media studies, I wanted to look at other actors who play a crucial role in that media ecology. While there has been much talk of the ordinary user becoming a 'produser' of participatory culture, other produsers remain absent in many respects from such discussions and discourse. In terms of advocating a software perspective on social networking sites especially, focusing on the alternatives and overlooked actors is imperative. In the final chapter of this dissertation, I turn away from my previous focus on Facebook, in order to look at another interface and other produsers of social media – the application programming interfaces (APIs) and the third-party developers. Given the crucial role that APIs and outside developers have played in the dissemination and growth of social networking sites, remarkably little attention has

⁴⁵ See Langlois et al. (2009a) for more on the notion of me-centricity and Facebook research.

been paid to these actors in previous research (for a notable exception see Cramer and Fuller, 2008; Langlois et al., 2009a; 2009b). Taking the micro-blogging service Twitter as my case, I look at how software signifies and is suggestive of things by examining the APIs as a location of power relations. In this sense, I find a qualitative and interview-based methodology to be an appropriate way of getting at some of the experiential and meaning-making capacities of software.

Secondly, I wanted to talk to third-party developers in order to ‘make sense’ of software in the context of Web 2.0. Approaching end users about their lived experiences with software might not be as fruitful, as social networking sites are not primarily experienced as software, but rather as media or communication platforms more generally. From a software studies perspective, what is of interest are the ways in which software as socio-material production has a profound impact on everyday life (Kitchin and Dodge, 2011: 13). A crucial site for such an inquiry into the sociality produced by software, I believe, are the originators and producers of software. Although the production of media is regarded as an important site of analysis within media studies, the cultures of code and programming have not yet gained much scholarly attention in social media studies.⁴⁶ What is needed is therefore a better understanding of the ways in which software participates not just in configuring end users, but also the lived experiences of programmers and developers, and how the assemblages of commercial Web 2.0 platforms and code underpins new forms of governance that both control and empower new creative opportunities for software production (Kitchin and Dodge, 2011: 111).

Informants and interview questions

Third-party developers are particularly interesting actors in the context of social networking sites, as they are both users and producers of software. Being in close contact with the underlying logic of the software of some of the currently largest media companies, third-party developers arguably have a unique position from which to reflect on and ‘know’ software. Unlike software engineers and developers actually

⁴⁶ Here it needs to be pointed out that there is one notable exception in regard to the lack of attention to cultures of code in cultural and media studies. This is the rich scholarship on free and open source movements. See for instance David Berry’s ‘Copy, Rip, Burn’ (2008) and Gabriella Coleman’s scholarship.

employed by these companies, third-party developers are free to talk about software and its perceived functionalities. Initially, I tried to get in touch with developers employed by Facebook or Twitter, but my attempts were largely unsuccessful. When I did receive replies, they made it clear that as developers, they were not allowed to disclose anything about the companies, let alone discuss technical details. When beginning to research the developer communities around Facebook and Twitter, it quickly became clear that the Twitter community was generally more open and accessible than for example the community surrounding the third-party development happening in conjunction with Facebook.⁴⁷ A combination of Twitter's long history of offering APIs for application development in contrast to Facebook's rather recent opening up of their API in 2009, as well as the existence of a very active mailing list for Twitter developers as opposed to the anonymised Facebook developer site, resulted in my choice to look more closely into the Twitter developer ecosystem. I became a member of the Twitter developer mailing list called 'Twitter Development Talk' (hosted on Google groups) in July 2010, and started recruiting informants from the list in August of the same year.

While I approached many more Twitter third-party developers and also received answers from more than 20 participants, what I will here refer to as a corpus of 20 informants consists of those developers who either got back to me with a more or less elaborate one-off answer to my questions, or with whom I entered into several rounds of questions and answers. Following the principles of grounded theory methodology (see Glaser and Strauss, 1968), my approach has been to gradually accumulate new data that built on the coding and interpretations from the initial round of data collection. Thus, the interviews follow a step-by-step process that offers the possibility to move back and forth between the various levels of analysis and data collection. The interviews were conducted mainly during two time periods, with the first set of e-mails sent back and forth during August and September 2010, and the second during May and June 2011. With the help of the Twitter developers' Google group, I gathered names and e-mail addresses of random developers participating in

⁴⁷ This inaccessibility of the Facebook third-party developers was not due to the developers themselves, but was related to the architecture and structural support of the developer forums hosted by Facebook. Whereas the user names of developers who were discussing issues concerning the Twitter API could in many cases be identified by a functional Gmail address, there was no easy way to identify and contact the Facebook developers.

the discussions and sent out an interview request. Most of the developers I approached responded fairly quickly, and many agreed to answer my questions. As most of the developers preferred to communicate with me via e-mail (as many of them were online much of the time anyway), the interviews were conducted using e-mail.

With the exception of two of the developers, the informants were all male. This was for the most part due to the fact that the overwhelming majority of participants who posted messages to the Twitter Development mailing lists were male. While I did not ask specifically for any background information from my informants, many of them disclosed information about their persona in the answers they gave. In terms of age, there were some significant variations, ranging from high school teenagers to people aged 50 or older. This age span also reflected the level of experience the informants had with computer software and programming. Whereas one of the high-school-aged developers I interviewed had for the most part taught himself web development languages such as PHP and scripting languages such as Javascript – both crucial languages in terms of Web 2.0 – the informants who grew up during the 1980s had learned languages such as BASIC and Logo. This is not to say that the younger informants were somehow less experienced than the informants who had been around software for quite a while, merely that they had different experiences. The age variation also reflected a variation in professional background, including (but were not limited to) students with high school- and/or college-level educations, young professionals with a web-development-related day job, but not necessarily one that involved the Twitter API, a PhD in biochemistry, the author of programming books, a university faculty member, CEOs of software companies, the project manager of a European Union project, an American visitor to CERN, and hobby app developers. The informants also differed in terms of their level of involvement in the third-party developer community. Some had only posted on the mailing list once at the time I contacted them to ask them to participate in my study, while others were very active members who posted up to several times a day during certain periods.

Initially, I sent out four to five sets of questions to the developers who agreed to participate, so as not to overwhelm them with too many questions or ask for too much. In many cases, the e-mail exchanges extended into several rounds of discussion that provided an opportunity to ask follow-up questions and to enter into discussions on topics that were of particular interest to individual informants. The questions I sent

out in the initial round were organised in three major themes or sets of questions. They covered a wide variety of issues, including programming practices and questions pertaining to software and the real-time web (see Table 1).

For example, a first set of questions concerned Twitter and the developers' personal interests in and motivations for using the Twitter APIs. I also asked more detailed questions about the nature of an API and the technical specifications of Twitter in order to learn about the technical issues behind software.

A second set of questions examined the developers' own experiences with programming in a Web 2.0 environment, and asked about how they got into programming. These questions were important for contextualising the developers' entry points into the social media industry, and for understanding their personal narratives and interests in programming, especially as they pertained to web programming and Twitter specifically.

A third set of questions sought to obtain the programmers' views on software. These asked them simply to explain the nature of software. In reviewing the literature on software studies, it quickly became clear that the concept of software is notoriously difficult to pinpoint. At the risk of asking a foolish or naïve question, I wanted to hear from the software producers themselves, and let them reflect on the nature of software and what it means to them. Interestingly, the answers I received differed as greatly as those to any other question.

A fourth set of questions, sent out to developers in August and September 2010, specifically addressed the theme of the real-time Web and temporality in relation to Twitter. After the first round of coding however, APIs and their role in social media, and the dynamics of the Twitter developer community crystallised as particularly important and interesting topics. I therefore decided to shift focus of my questions from the real-time Web to the theme of APIs in the second interview period, in 2011. While questions concerning the impact and role of APIs were also brought up in relation to programming practices and software, APIs and the developer ecology became the core focus of most of the interviews, and were included in some manner in all the question sets.

The interviews were already fully transcribed and were coded and analysed in accordance with the grounded theory approach (Glaser and Strauss, 1968). The

coding scheme, comprised of categories and sub-categories, was developed using open and axial coding.⁴⁸ As the transcriptions were already quite structured, I decided to do the coding manually, as opposed to using software. I printed out all the interviews and began to identify and name phenomena found in the transcripts by writing notes in the margins. I repeated this process two or three times for each interview, at different points in time, iteratively refining the codes to comprise a more stabilised coding scheme. Finally, these axial codes were analysed to identify the core themes in the data sets. When writing Chapter 8, I frequently went back to the interviews and carefully chose quotes carefully, in many instances refining the code labels in their entirety.

Sample of interview questions:

Twitter

- What is your interest in using the Twitter API?
- What kind of projects are you interested in developing using the Twitter API?
- Could you tell me a bit more about the technology behind Twitter? How would you explain its main technical features?

Programming practice

- What kind of programming skills do you have, or think is needed in web programming today?
- How did you get into programming?
- In what ways do you think programming and the job of programmers have changed with the introduction of social media, if at all?
- Could you elaborate on how you go about writing code? What are the typical steps you take when developing an application using the Twitter API for example?

Software

- How would you describe software to someone who is not a programmer?
- There is a new academic field called software studies, which tries to study software from a humanities perspective, as something that has a profound impact on culture and society. Does it make sense to you to “understand” software in these terms? What comes to your mind when thinking about the

⁴⁸ Analysis of data in Grounded Theory methodology follows two or three main stages: open, axial, and selective coding. Open coding refers to the initial stages of making sense of the data by naming, labelling, and categorising the data. In axial coding, the data is put together in new ways by making connections between the categories identified in the open coding.

<p>impact of software on everyday life?</p> <ul style="list-style-type: none"> • Some people say that software is simply the same as the source code. Would you agree, or what in your opinion defines the nature of software? <p>APIs and community</p> <ul style="list-style-type: none"> • How would you describe the impact of APIs and what they have done for developers and programmers? • What exactly is an API? How would you define an API? • To what extent do web programmers interact more with APIs than with ‘backend’ programming? • I’ve noticed there is a thriving Twitter developer community out there, especially in the Google group. How important is this community to you? • How important do you think third-party developers have been to Twitter’s success?

Table 1 Sample of interview questions

Reflections on using e-mail interviews

The purpose of qualitative interviews is to obtain a more in-depth and nuanced understanding of the phenomena being studied (Kvale, 1996; Rubin and Rubin, 1995). Qualitative research interviews can take different forms, from structured to loose and informal conversations. In my case, conducting more or less structured e-mail interviews proved to be a purposeful method for gaining a better understanding of coders and coding practices in the context of social networking sites. As all the informants recruited were based abroad, mainly in the United States, but also the UK and other European countries, conducting the interviews online seemed like the most viable option. When approaching my informants, I quickly learned that e-mail was the preferred medium for these conversations. Given that my informants were programmers who often spent a considerable amount of their time in front of the computer, e-mail was a quick and efficient way to talk/write about their experiences with code and coding. Conducting the interviews using e-mail allowed participants more time to think through and reflect upon the issues that were being addressed, and to tailor the conversations and subsequent e-mails to the interests and previous answers of the individual informant. Of course, other media could have been used for the interviews, such as the telephone, video calls through Skype or instant messaging. The combination of ease, flexibility on behalf of both interviewee and interviewer, the

advantage of e-mails as transcriptions, and the informants' clear preference for e-mail provided the main reasons for my choice of e-mail as my method of communication.

It is not always clear that using e-mail to ask a few questions should be considered an interview, and not for instance a web survey. Following Edgar Burns (2010), who has written on e-mail interviews as qualitative research, we can say that while a web survey takes on a more or less standardised format with the same questions sent to a large number of people, e-mails can better extend the one-off event of a survey, and be personalised and tailored to the specific background and interests of the informant. E-mails, in contrast to a web survey, allow for multiple rounds of questions and follow-ups.

While there are many obvious benefits with using e-mail interviews, there are many limitations as well. First of all, the quality and length of answers are not always as 'good', in the sense that the answers provided can be quite brief. In many cases, I did not receive as many elaborate answers as I would have wished. While nearly everyone I contacted got back to me with some kind of response, the length and quality of the individual answers varied greatly. Sometimes the respondents would answer by writing only one or two sentences, while others would write several lengthy paragraphs. In addition, conducting the interviews via e-mail allowed for only a limited opportunity to turn the interviews in a desired direction. In this kind of interview context there were no opportunities to interrupt or adequately follow up by probing for more information during the answering process. While I sent out many follow up questions, this was not a guarantee that the questions were really answered. In some cases, the developers would provide lengthy and elaborate answers that were quite technical in nature, often containing information that could also have been easily acquired by a regular Internet search. Indeed, some developers answered questions about technical specifications by simply copy and pasting from a Wikipedia article. However, as I pointed out earlier, the sample that consists of twenty informants does not include those respondents, who only answered in a very general or brief manner.

Burns (2010) points out another drawback with using e-mail for interviews. This pertains to the fact that '[e]mail interviews do not provide body language and other contextual cues for the interviewer'. There is also the potential risk of miscommunication and misunderstandings, due to the fact that the respondents cannot ask for instant clarification on the questions posed. As such, conducting e-mail

interviews requires a great deal from the interviewer in terms of how to pose the questions. The questions need to be understandable, precise, and clear. In cases where I attempted to ask more complicated questions that were not as straightforward, I needed to think about how to contextualise them. For example, how do you contextualise ontological questions about the nature of software without scaring off the developers, or simply receiving a copy-pasted Wikipedia answer? In some cases, I started off by providing more context in order to make sure that the respondents had a better chance to understand what I was trying to get at. For example, because I was interested in having a conversation about software from the viewpoint of developers, I introduced one of my questions by providing some general background about the notion of software studies. Only then could I ask how they felt about software or would explain it to a non-programmer. Another way to contextualise questions about the nature of software was to first pose a problem or apparent contradiction and ask them for their own opinions or solutions. For example, one interesting theme that emerged was the distinction between source code and software in general. In order to get the programmers to ponder this distinction, I would pose the question in terms of a contradiction, and ask them for their own view or opinion on whether software could be reduced to its source code.

Despite the obvious shortcomings of e-mail as an interview method, I believe using e-mail to collect information and personal narratives from well-placed sources (informants) offers an appropriate methodological approach to gain important insight into the field of study. Especially in terms of interviewing programmers, e-mail turned out to be a very useful means of communication, as this is a group of people already very accustomed to communicating electronically. As it turned out, the medium of e-mail was just the right choice for talking to programmers, as many were most comfortable with using this written form of communication and appreciated the flexibility it provided in terms of being able to take one's time to answer and respond at their own convenience. Importantly, interviewing informants about their own thoughts and experiences with programming and software, showed how software becomes meaningful, how it has the capacity to produce meaning in different contexts.

Concluding remarks

In the end, doing technography as understood in the context of this dissertation is all about specifying how sociality is constructed and sustained in and through coded objects and processes. Technography then, is seen as a methodology appropriate for studying how technical structures and mechanisms embody the potential for shaping collective associations and cultural outcomes.

In the next three chapters I turn to the concrete case of Facebook, demonstrating the usefulness of the technographic approach. I will be concerned with how the software processes and mechanisms underlying the platform actually work and what the operational logic of specific protocols and algorithms are, in order to analyse how these coded architectures can be said to be suggestive of specific ways of assembling and organizing sociality.

In the next chapter I turn to the ways in which the Facebook platform has changed over the years, focusing particularly on the technical modes of connectivity manifest in Facebook's Open Graph protocol. Through a detailed reading of the specific affordances of the protocol and the practices it engenders, I will focus on the technosocial structure of Facebook as an attention apparatus.

Chapter 5. Open Graph protocol: Arranging attention

Facebook is currently the second-most accessed web site in the world.⁴⁹ Its mission statement is to help users connect and share with the people in their lives. Facebook is not just a helping hand though; it is a business that hinges on attracting, capturing, and keeping attention for the sake of commercial profit. While Facebook certainly enables connections to be forged by facilitating a space for users to come together and share the things they care about, it is important to keep in mind that these are simultaneously *engineered connections* and spaces of *designed experiences* (see van Dijck, 2012). In this first investigation into programmed sociality, it is important to develop a critical understanding of the coded means and technical mechanisms through which this connectivity is shaped on Facebook for the purpose of capturing a certain kind of attention – that of participation.

In this chapter, I develop an understanding of what I call a *technicity of attention* in social networking sites. I hold that these sites treat attention not as a property of human cognition exclusively, but rather as a sociotechnical construct that emerges out of the governmental power of software. Specifically, I take the Facebook platform as a case in point, and analyse key components of the Facebook infrastructure, including its Open Graph protocol and its ranking and aggregation algorithms, as specific implementations of an ‘attention economy’. Here, my understanding of attention economy is informed by the etymology of the word *economy*, as ‘household management’, in order to illuminate a discussion on the ways in which attention is organised and managed within a localised mediated context.

My aim is to take a step back from the proliferating anxiety-ridden discourse on attention and the media, a discourse that has recently emerged as part of the so-called ‘neurological turn’ (see Carr, 2010; Wolf, 2007).⁵⁰ The use of recent neurological

⁴⁹ See Alexa.com, last checked April 24, 2012.

⁵⁰ See Anna Munster’s (2011) ‘Nerves of data: the neurological turn in/against networked media’ for a good discussion on the neurological turn. Writers like Nicholas Carr have been quite visible within popular media discourse on the topic of attention during recent years, presenting a somewhat technologically determinist argument about a loss of concentration being the result of an increased level of multitasking generated by networked media.

research to justify arguments about an apparent shift in cognitive capacities brought about by new information technology has recently emerged as a clear tendency among critical media theorists and philosophers (see Hayles, 2007; Stiegler, 2010). While these efforts make a considerable contribution to a humanistic understanding of the interrelation between the brain and behaviour, the micropolitics of power involved in the media are often sidestepped as an object of analysis in favour of the impetus of e.g. fMRI scans. In this chapter, I therefore counteract this tendency by focusing on the specific algorithmic and protocological mechanisms of Facebook, looking at the ways in which they enable, shape, and induce attention in conjunction with users.

This chapter addresses the capacity of attention not as a type of spectatorship, but rather as a mode of participation that is subject to a form of ‘governmentality’ (Foucault, 1991). According to Foucault, governmentality refers to the rationalities that underlie the ‘techniques and procedures for directing human behavior’ (Foucault, 1997: 81). Following work on the ‘technological’ aspects of government (see Lemke, 2001; Miller and Rose, 2008), this chapter provides an account of how Facebook operates as an implementation of an attention economy directed at governing modes of participation within the system.

Specific governments may entail different rationalities used to guide the conduct of people (see Foucault, 2007). Building on this idea, I argue that the technical rationalities – what I here refer to as *technicity* – used to govern participation on Facebook are realised in at least three different ways. These organising principles can broadly be said to correspond to: 1) an automated, 2) an anticipatory, and 3) a personalised way of operating the implementation of an attention economy on Facebook.

My analysis of Facebook focuses on the specific infrastructural arrangement of participation, as materialised in the Open Graph protocol and underlying algorithms. To do so, I rely on the technographic approach outlined in the previous chapter. This involves a reading of the technical inscriptions and affordances involved in the composition of the Facebook infrastructure. I draw on both specific technical documents and popular commentary related to the Open Graph protocol and algorithmic logic of Facebook as a way of grounding the analysis in the specificities of the software medium itself.

The goal of this chapter is to explore how software, through protocols and algorithms, has the capacity to govern and manage users. I contend that the concept of technicity provides the media theorist a way in which to understand how specific material arrangements such as Facebook enable, capture, and augment awareness and participation relative to users.

Paying attention to attention

The topic of attention has always been important for media and communication research. In fact, one may claim that in many ways the field of media studies developed as a direct consequence of academic concerns over the relations between media technology and attention. Questions concerning how attention is captured, formed, and retained have been an important part of media theory since at least the mode of ‘distraction’ was identified as central to understanding cinema as a mass cultural art form (Benjamin, 1999; Kracauer, 1995). As a result, attention has been a key theme in everything from film and television research (see Dayan, 2009; Newman, 2010) and news and journalism to public relations (Bernay, 1947).⁵¹

There is a tendency within media studies to conceptualise attention solely as a faculty of perception, where attention is more or less reduced to the notion of visual attention (Wise, 2012:165). This emphasis on perception and visual attention arguably stems from the ways in which attention is commonly understood in the psychological and cognitive sense. According to psychologist Harold Pashler:

Two primary themes or aspects characterize the phenomena people allude to with the term attention: selectivity and capacity limitation [...] One is that conscious perception is always selective. Everyone seems to agree that, at any given moment, their awareness encompasses only a tiny proportion of the stimuli impinging on their sensory systems [...] The second phenomenon to which causal usage of “attention” alludes is our limited ability to carry out various mental operations at the same time’ (1998: 2).

In line with Pashler’s characterisation, attention is widely understood as the process of *selecting* information for further mental processing, and/or as a mental capacity that

⁵¹ In terms of television research, Michael Newman offers an interesting historical account of how discourses of attention have played a key role in forming both the agenda for much of reception studies as well as had very real implications on the programming practices of television production. A key concern within the literature identified by Newman, has been the variability of ‘attention span’ by age.

describes the degree of *focus* directed at something. As such, it is perhaps not surprising that discourse on media and attention in the age of the Internet, with its hyperlinked structure and abundance of available information, has been concerned with how networked power may lead to a lessening of the cognitive capacity for directing attention. It is specifically in regard to the Internet that findings from recent neurological research have been brought to bear, with N. Kathrine Hayles' claiming that these show there to be a 'generational shift in cognitive styles' between 'deep' attention and 'hyper' attention (2007: 187).⁵² The ubiquity of information enabled by the Internet, coupled with the notion of the brain as a limited information-processing machine, has furthermore provided the basis for what some scholars have described as a new type of *attention economy* (Goldhaber, 1997; Franck, 1998).

The notion of the attention economy has been used to designate the increased competition for people's attention in an age of information overload. When information increases, the attention to make sense of it decreases. Given this kind of information environment, the fight for consumers and their attention becomes more critical. As a result of a complex market system where scarcity is not bound to money, but rather to time and attention, efforts to attract this currency intensify. With a few notable exceptions (e.g. Lazzarato, 2006; Lanham, 2006), media research on attention has adopted the prevailing and dominant views on attention as they are articulated within psychology and neuroscience.

The question lingers, therefore, of how a conceptualisation of attention as endowed with value, distributed and organised (as 'the attention economy') would operate, if we as media researchers were to start with the medium itself, as opposed to taking our point of departure in what fMRI scans tell us about the brain. By shifting the focus in this manner, I do not mean to say that humanities scholars should be unconcerned with the ongoing developments within cognitive science and psychology. Nor do I wish to deny that attention should be seen as a property of human cognition. Rather, I

⁵² Hayles makes the distinction between the two types of attention in the following manner: 'Deep attention, the cognitive style traditionally associated with the humanities, is characterized by concentrating on a single object for long periods (say, a novel by Dickens), ignoring outside stimuli while so engaged, preferring a single information stream, and having a high tolerance for long focus times. Hyper attention, by contrast, is characterized by switching focus rapidly between different tasks, preferring multiple information streams, seeking a high level of stimulation, and having a low tolerance for boredom' (2007:1).

suggest that something is needed to complement the notion of attention as a purely cognitive property, and note the lack of an analytical tool that may help us understand how attention is rooted in and constrained by the medium itself. To fill this gap, I offer an account of *technicity*, understood as the ‘productive power of technologies to make things happen, in conjunction with people’ (Kitchin and Dodge, 2011) and as the ‘coconstitutive milieu of relations between the human and their technical supports’ (Crogan and Kennedy, 2009: 109),⁵³ as a means of analysing attention as it arises out of the software-subject continuum in the context of Facebook.

In this chapter, I will therefore not answer the question of what attention *is*, but rather show how software has the capacity produce and instantiate modes of attention, specific to the environment in which it operates. Importantly, though, the productive power of technology, as it is signified by the concept of technicity, does not operate in isolation or as a unidirectional force. It should, rather, be understood as a relational force. Taking such a perspective allows the theorist to see attention as an emerging property of sociotechnical relations.

Arranging attention: The case of the Facebook platform

In 2007, Facebook launched the ‘Facebook platform’. Access to valuable user data was provided, and third-party developers were offered the opportunity to deeply integrate with the Facebook website. In many ways, the launch of the platform signalled a first step towards Facebook not merely becoming the most popular online social networking service, but also as a model for the infrastructure of the social web itself. By opening up its core to applications, Facebook provided access to what it calls the *social graph* – ‘the real connections people have’ (Geminder, 2007). During the second *f8*, Facebook’s annual developer convention, in 2008, Facebook introduced ‘Facebook Connect’, a product that prompts the expansion of users’ Facebook identities to other parts of the web. Facebook Connect made it possible for users to register on external websites using their Facebook ID. In a telling press-release entitled ‘Facebook expands power of platform across the web and around the world’, Facebook Connect was presented as a way to leverage the power of

⁵³ I am not explicitly using technicity in the sense of ‘originary technicity’ as Derrida and Stiegler do, although the position I am taking with regard to software is sympathetic to the metaphysical view on the aporetic relationship between humans and technology.

information feeds (Facebook, 2008). By connecting Facebook to external websites, users' actions on these websites (provided they had used their Facebook account to sign in) would subsequently be fed back to Facebook as data.

Subsequently, during the fourth *f8* in 2010, Facebook released what Chief Technology Officer Bret Taylor recently described as the most profound change to their platform since its launch in 2007 – the *Open Graph* (Taylor, 2011).⁵⁴ Designed to facilitate connections between people and things, Open Graph consists of a protocol, an application programming interface (API), and social plug-ins, including the now-pervasive 'Like' button. The Open Graph protocol describes a way to build a semantic map of the Internet. Technically, Open Graph is implemented through RDFa, a data model for mapping that allows webpage owners and application developers to mark up human-readable data with machine-readable indicators (see <http://ogp.me>). For instance, a website such as the Internet Movie Database (<http://www.imdb.com>) can be semantically linked up to Facebook's core service by adding some metatags into the html of the imdb site. This mark-up code turns external websites and digital objects into Facebook graph objects.

The Open Graph protocol allows Facebook to track and process user data across the web with the use of social plug-ins. These plug-ins function as small 'hooks' that provide meaning to nodes (i.e. webpages, movies, books etc.) that were not meaningful to Facebook before being linked to the larger infrastructure provided by Open Graph. With the help of these hooks, all kinds of entities can be given a social network presence. The purpose of the Open Graph is thus to create links between various nodes that extend beyond the core site of Facebook.com, whether the link is established between two users or between a user and a webpage. Importantly, this map of connections allows the data that flows between actors to be tracked and processed more easily, since all the data created in and through these articulations is fed back to Facebook.com. Essentially, Open Graph constitutes a centralised infrastructure that generates value by decentralising social action. This data-intensive infrastructure is powered by the implementation of 'Like' buttons, which allows for the registration and tracking of user actions tied to external sites. The 'Like' button

⁵⁴ More information about the Open Graph protocol and its specifications can be found at <http://ogp.me> and <http://developers.facebook.com/docs/opengraph>

thus functions as a storage and transmission device, capturing the attentional data of user activity.

What happens when a user clicks the ‘Like’ button? First, a connection between two nodes in the graph is established. In the case of the movie on IMDB, this articulation occurs between the Facebook node and the movie node. Subsequently, this tiny piece of interaction is made visible or potentially visible in a variety of ways, and in many different locations. Let us say that I ‘like’ the movie *The Matrix* on imdb.com. The action of liking gets translated into a piece of data on Facebook, where *The Matrix* is now archived as one of my favourite movies. In addition, the action of ‘liking’ generates a story on my personal wall saying ‘Taina likes a link’, along with a post of the actual link and picture of the movie. User actions thus follow the user, as opposed to the actual webpage on imdb.com, for example. This is important in at least two ways: it makes it possible to aggregate data about the user, and provides a persistent link between the user and her favoured concepts, whenever such concepts appear on a webpage. As Bret Taylor of Facebook suggests, ‘the Like button offers users a lightweight and consistent way to share the things and topics that interest them’ (2010). Clicking the ‘Like’ button thus signifies approval of something — a convenient way for users to tell their networks that a particular piece of content is worth their attention. ‘Liking’ things across the web has thus become an important and time-efficient way of doing ‘identity work’, an easy way to show who the user is and what he or she cares about. The fact that ‘liking’ collapses all affects into one metric tied to the Facebook ID of a particular user, makes it possible to aggregate valuable data for advertisers, and for Facebook to learn as much about the user as possible.

In the attention economy of the web as a whole, where user attention is a valuable commodity due to the vast information and products available (Goldhaber, 1997), Facebook has become a key actor in the competition for ‘eyeballs’. As Facebook suggests, ‘the average media site integrated with Facebook (with the ‘Like’ button or other plug-ins) has seen a 300% increase in referral traffic’ (Sullivan, 2011). A crucial concept for understanding the idea behind Open Graph is what Facebook refers to as the *social context*. ‘Social contexts’ are the people and friends who have already interacted with a piece of content that a user is interacting with. Through the Open Graph, Facebook makes it possible for external websites and brands to socially

contextualise the content they display. So when visiting IMDB or another website that has integrated with the Open Graph, users will be able to see how many people have liked it or recommended the content, and which of his or her friends have also like it.

EVERYONE RECENTLY LIKED

Barry Gaynor likes **Trucker Jacket – Stonewash**
The Levi's® classic Trucker jacket. The relaxed fit has front and back yokes, vertical seaming and a spread collar. Two button-through flap pockets (one trimmed with our signature Red Tab™) and two hand-warmer pockets in front.
4 hours ago · [See It](#) · [Buy It](#)
[Like](#) 53 people like this. Be the first of your friends.

Barry Gaynor likes **Vintage Trucker – Dark Rigid**
This rugged, richly dyed denim Trucker is built to stand up to the wear and tear of the road. Full-length sleeves, a waist that sits right above the hip, classic brass buttons and slightly deconstructed seams make this a jacket fit for the long-haul.
4 hours ago · [See It](#) · [Buy It](#)
[Like](#) 157 people like this. Be the first of your friends.

長谷川秀樹 likes **Super Skinny 510™ Jeans – Reef**
Our slimmest fit from hip to hem, the Super Skinny 510™ Jean sits below the waist like our Skinny 511™ Jean, then gets extra narrow through the leg -- all the way to the ankle. A bit of stretch in the denim keeps it comfortable.
5 hours ago · [See It](#) · [Buy It](#)
[Like](#) 509 people like this. Be the first of your friends.

Keejong Lee likes **Slim Straight 514™ Jeans – 3D Coated**
A bestseller for its universal slim fit, which never feels too snug or looks too baggy. Cut low at the waist and close through the seat and thigh. Straight leg.
7 hours ago · [See It](#) · [Buy It](#)
[Like](#) 1,328 people like this. Be the first of your friends.

Figure 1 Levi's online store. Screen shot from May 9, 2010.

According to Facebook, when social context is provided, the amount of engagement goes up dramatically (Facebook Engineering, 2011). For example, the jeans brand Levi's, one of the first brands to integrate with the Open Graph, has apparently seen a '40 times increase in referral traffic from Facebook after implementing the 'Like' button in April 2010' (Sullivan, 2011). In the type of attention economy promoted by Facebook, the value of information is based on friends. As Gerlitz and Helmond (2011) have argued, there seems to be a move away from the link economy based on

the authority of links regulated by search engines, to a ‘Like economy’ regulated by the wisdom of friends. While Facebook has become an important mediator for brands in the competition for attention, the way Facebook has positioned itself is less about the quantity of impressions, or ‘eyeballs’ than the apparent ongoing engagement of recognisable users and their network of friends. I will later return to this notion when discussing what I argue constitutes a move from a public to a personalised attention economy.

The Facebook platform constitutes an ‘assemblage’, in the sense that it brings together various heterogeneous elements to produce and distribute flows of attention (see Deleuze and Guattari, 1987). The Open Graph protocol, the API, and the ‘Like’ button cannot function effectively independently of each other. Together, however, these infrastructural and medium-specific elements provide a foundation for the organisation and measurement of users and their connections. Seen in this way, the Facebook platform has become an infrastructure that works invisibly in the background to shape forms of social activity. As Edwards points out, infrastructures always ‘promote some interests at the expense of others’ (2002: 191). I argue that the infrastructure of Facebook is built around the logic of creating, capturing, and processing attention. The entire system hinges on attention in a very specific way. Facebook not only captures attention in terms of recording already-existing attentional data embedded in links and clicks, but also creates and ‘interferes’ in the production of attention by suggesting the types of content that deserve users’ attention.

The most important repercussion of the ‘Like’ button lies in the potential of the ‘like’ action to become visible on other users’ News Feeds. If a user’s action of ‘liking’ something on or off Facebook is algorithmically calculated as relevant, it will appear in the Facebook News Feeds of some of that user’s friends. Not everything that is ‘liked’ across the web actually makes it on to users’ News Feeds. The sorting mechanism that ultimately decides *what* is to be shown on Facebook users’ News Feeds and *when* it is to be shown, is the *EdgeRank* algorithm. Filters have, of course, long been integral to the management of information and attention through information technology. The psychologist Herbert Simon, who coined the term ‘attention economy’ in a 1971 paper, suggests that the design goal of information processing systems should always be to provide users with only the information they

need to know, a principle that in turn emanated from the decision-making under pressure paradigm of early cognitive simulation (see Crogan, 2011).

This is the programmatic purpose of the EdgeRank algorithm. It selects and ranks the information that it calculates users need to know. Herein lies the power of the algorithm in a digitally-mediated culture: to ‘enhance’ the plethora of collected data in order to ‘identify patterns and trends’, and to use this information to ‘profile, model, predict and simulate people and situations’ (Kitchin and Dodge, 2011:103). EdgeRank augments, supports, and governs attention by simulating the cognitive function of attention as a sense-selecting mechanism. The ‘art of government’ underlying EdgeRank can be said to hinge on what Rancière has called the ‘distribution of the sensible [...] defining what is visible or not in a common space’ (2004: 12-13). According to Rancière, ‘the distribution of the sensible reveals who can have a share in what is common to the community based on what they do and on the time and space in which this activity is performed’ (2004: 12). Seeing algorithms in this way points to the ways in which software embodies a politics (see Winner, 1986). Thus, the politics of algorithms and their governmental power refers to the ways in which algorithms are ‘making decisions [...] about who to deal with and how to deal with them’ (Beer, 2009: 989). The power of EdgeRank, then, pertains to its gatekeeping function, it decides what information to present to which user, and in what ways.

As the name suggests, EdgeRank is a ranking algorithm designed to pass judgement on the relative importance of ‘edges’, understood as the links between two nodes in the graph. The higher the algorithm ranks a piece of content in terms of significance, the more visible that shared content becomes (see Tonkelwitz, 2011). Attention can thus be encoded as information. As users continuously connect with new nodes (i.e. photos, movies, webpages, other users), novel connections are forged on an ongoing basis. By updating the social graph – that is, by adding connections within the schema – all sorts of activity happens on Facebook. When a user comments on a friend’s photo, for example, the user makes a connection or generates an ‘edge’ from him or herself to the friend’s photo. Evidently, News Feed only displays a subset of stories (or edges).

Every time connections are forged, Facebook assigns a value to that ‘edge’, to determine whether or not it should be displayed in a friend’s News Feed. This value is based on three main factors: affinity, weight, and time decay (see the next chapter for

a more detailed description on the logic of EdgeRank). These values are assigned to units of information and the forms of interactions taking place within the auspices of Facebook, to determine how interesting the story would potentially be for particular users. For instance, if one user frequently checks another user's profile, the 'affinity' between the former and the latter is ranked higher. This is applied in the calculations of the algorithm as it generates the first user's News Feed and makes it more likely that the second user will feature within that feed. 'Weight' refers to the type of 'edge', whether the connection established between two nodes is a 'like', a comment on someone's photo, or an uploaded video.

Social media marketing firms have suggested that Facebook 'weighs' comments as more substantial than a 'like', and that visual media content including photos or videos is more strongly weighted by the algorithm (Walter, 2011). 'Time decay' simply refers to the age of the 'edge'. As with other modes of attention economy, such as the curation of search results, Facebook regulates access to information based on 'relevance' metrics that prioritise popularity. However, while the authority of a link on Google is shaped by the accumulation of inbound links, and as such depends on public attention, popularity measures such as 'affinity' in EdgeRank are dependent on the programmatic profile of particular users. For instance, if a user does not show interest in the pictures of a particular Facebook friend, the system will assume that the user has no interest in this contact at all.

During the *f8* conference in September 2011, Facebook introduced further and significant changes to the platform and the ways in which attention is organised. These changes included a completely new profile design called the 'Timeline'; a real-time feed called 'Ticker'; an updated version of the Open Graph protocol, with a greater emphasis on applications; and a new algorithm called 'GraphRank', responsible for managing user interaction with applications. Whereas the 2010 version of the Open Graph allowed for connecting to the rest of the web by 'liking' it, the updated version apparently allows for connecting to 'anything one wants in any way one wants', in the words of Facebook founder Mark Zuckerberg (2011). No longer confined to 'liking' things, Facebook introduced new verbs such as 'read', 'watch', and 'listen' into its logic. This allows users to share the fact that they are listening to music or reading an article, instead of simply sharing something they 'like'.

The most significant change, however, lies in what Facebook calls ‘frictionless sharing’. No longer must users explicitly click the ‘Like’ and ‘Share’ buttons, or copy and paste links into their status updates. Frictionless sharing refers to authorising a Facebook ‘app’ only once in order to let it automatically share a user’s interactions with it every subsequent time he or she ‘reads’, ‘listens’, ‘run’, ‘watches’, ‘cooks’ etc. using that particular app.

Through ‘Ticker’, the real-time feed on the right-hand column of the Facebook homepage, users are now able to see a constant stream of information about what their friends are currently doing on Facebook or with Facebook-connected apps. For example, as a result of the Open Graph-enabled ‘frictionless sharing’, every time a user listens to music using the Spotify application (and has authorised the app to communicate with Facebook) a story is published on Ticker informing that user’s friends that ‘X is listening to Y’.

With a new emphasis on seamless application integration with the Facebook platform, new kinds of activities are increasingly given priority and weight on the News Feed. In order to sift through and organise the proliferating user activity with apps, Facebook introduced the GraphRank algorithm, which is responsible for measuring and finding all the ‘interesting patterns’ that emerge from the uses of apps. The function of GraphRank ‘is to figure out what activity is most interesting’ to a particular user (Taylor, 2011). Adding to the already heavily-personalised News Feed, ‘GraphRank is designed to give more prominence to engaging activity [...] GraphRank isn’t a global score, but a personalized view of you and your friends’ taste’ (Taylor, 2011). GraphRank is specifically tailored to aggregate app interactions and display these through various summaries or reports.

While Facebook remains interested in users’ specific interactions with objects, GraphRank as an attention economy shifts the focus away from a ‘Like’-centric and conventional object-oriented attention economy towards a trend-centric and anticipatory attention economy. In order to elaborate on this and the increased customisation and personalisation of attention, I now turn to a conceptualisation of ‘technicity’ as a means to help frame an understanding of how Facebook’s infrastructure articulates a specific attention economy.

A technicity of attention

Media technologies play a crucial role in the formation of attention, both enabling and constraining awareness. As the philosopher Bernard Stiegler suggests: ‘whatever a given society’s form may be, one of its most distinctive features is the way in which it forms attention’ (2010: 19). Digital media have profoundly broadened the scope of awareness in terms of expanding users’ spatiotemporal registers. To speak of the technicity of attention offers a way to view the ways in which processes and practices of attention are grounded in a sociotechnical milieu. The concept of technicity has most prominently been developed in the continental philosophical tradition by Martin Heidegger, Gilbert Simondon, and Bernard Stiegler respectively. As a result, as James Ash points out, technicity has at least three meanings: ‘as a persuasive logic for thinking about the world; as a mode of existence of technical objects; or as an originary condition for human life itself’ (Ash, 2012: 189).

Technicity becomes a useful concept for an understanding of attention in digital culture, as it captures the power of articulating technologies and users in ‘localized assemblages of practices’ (Mackenzie, 2002: 12). Technicity understood as a kind of ‘technical mentality’ (see Massumi, 2009; Simondon, 2009) points to the ways in which technologies embody ‘mentalities’ or modes of framing the relations between living and nonliving processes in order to achieve certain ends. Here, I suggest furthermore that technicity can be thought of as a mode of governmentality that pertains to technologies. How then can we begin to understand the articulation of software and users on Facebook as productive of governing attention in particular ways? I will turn to this question in the remainder of this chapter by focusing on what I take to be three particular characteristics of the technicity of attention on Facebook: automated attention capture, a way of managing attention that is anticipatory in nature, and a move from ‘public’ to ‘personalised’ attention economies.

Automated attention management

Increasingly, the techniques put in place to assign meaningful value to information operate on the level of what Nigel Thrift (2004, after Clough, 2000) has called the ‘technological unconscious’. This form of unconsciousness, can be understood as the powerful operations of software putting its mark on the conditions of existence, where

living and nonliving processes are increasingly being programmatically addressed, correlated, and anticipated in unseen and unknowable ways. This unconsciousness is however not to be understood as imaginary, but rather in terms of the actual computational processes that run in the background, beneath and beyond what is perceivable to users via the interfaces of the computer.

On Facebook, software sits in the background ‘paying attention’ to user activity. It records, stores, and processes the data, constantly tweaking its models relative to how the data and hence the graph changes. The ways in which software is increasingly employed to process data in near real-time in order to ‘distribute the sensible’ - defining what is visible or not in a common space – can be seen as a mode of governmentality based on ‘automated management’ (Kitchin and Dodge, 2011). According to Kitchin and Dodge, automated management is the regulation of people and objects through processes that are automated, automatic, and autonomous (2011: 85). Faced with an algorithmically-sorted social networking system like Facebook, users do not merely browse the content that *they* find interesting; the ‘interesting’ content increasingly finds them.

The development of the Facebook platform, beginning with the ‘Like’ button and fully actualised with GraphRank, paradoxically makes social media less participatory through the notion of ‘frictionless sharing’. When the user no longer must explicitly push buttons, paying attention shifts from an active to a more passive mode. In this sense, one could argue that the conventional object-oriented way of conceptualising attention as a cognitive capacity directed at a specific object needs to be rethought, in an age of digital culture where software is capable of registering massive amounts of behavioural data ‘without any active involvement, decision to initiate or even awareness on our part’ (Hansen, 2012: 53). The apparent shift from spectatorship to participation as a measure of attention need not even be tied to intentional or active participation as in explicitly clicking a ‘Like’ button. Rather, as the development of the Open Graph for apps and ‘frictionless sharing’ attests, every potential user action is turned into an attention signal, marking a shift from an attention economy based on active/explicit to passive/implicit participation. The new Open Graph protocol creates an attention economy based on the leveraging of users’ passively-created activity data. As more and more services become connected to Facebook – through easy-to-

add HTML mark-up, for example – life becomes ever more measurable and thus governable.

Anticipating attention

Through means of automated management, Facebook is always already oriented towards the future. As Facebook-engineer Bosworth (2007) suggests, in a rather anthropomorphic way, the News Feed algorithm sifts

[...] through the enormous volumes of information about our friends on Facebook and picks just the best pieces to show us. While we eat it is keeping track of whom we seem to be keeping an eye on recently as well as remembering whom we have cared about in the past [...] it needs the information to do a better job picking stories because it thrives on people finding its stories useful and entertaining.

It is this always already anticipatory logic of the decision-making software that significantly drives the technicity of attention in digital culture.

According to Bernard Stiegler, anticipation is formed through the relationship between present and past experiences that have been externalised into specific material forms (2010: 18). The data-driven logic of Facebook makes the platform what Stiegler terms a ‘mnemotechnic’ – a form of ‘technical remembering’ – that allows for a mode of government that has the capacity to take into account the probability of subjects’ actions (see Stiegler, 1998; Lazzarato, 2007). Algorithms are anticipatory in their very ‘operational logic’ (Wardrip-Fruin, 2009), meaning that anticipation is inscribed into the very mechanics and rules of the system, as evidenced by the basic control-flow statement used to set up the calculable pathways. A program will execute a certain section of code only if certain conditions are met. Otherwise, it takes an alternative route, which implies that particular future circumstances are already anticipated by the conditional construct of the if-then statement. The ways in which algorithms operate is thus reminiscent of what Ben Anderson calls an ‘anticipatory action’, which constitutes a ‘seemingly paradoxical process whereby a future becomes cause and justification for some form of action in the here and now’ (2010: 778). Importantly, the way in which attention is managed on Facebook is not just anticipatory, in prompting participation, but rather a form of self-perpetuating anticipatory action that seeks to realise its own future.

In this sense, the introduction of GraphRank as the rationale for an attention economy could be seen as a self-fulfilling prophecy, as it seeks to actualise its predictions through its presentation and mechanisms to reward participation. The GraphRank algorithm continuously surveys users' interactions with Facebook-enabled apps in order to find the most interesting patterns. Once these patterns are found, they are fed back to users via the News Feed. Consequently, even more users will apparently act in the way that the algorithm predicts. The operational logic of the Facebook algorithms thereby works endlessly to produce a desired social order that these algorithms have themselves predicted in the first place. Attention is both a measure for predicting the future and something that prevents another future from happening. Every action taken by users on Facebook or Facebook-connected apps contributes to support of the trend-centric logic of Facebook. For Facebook, it is not the one song you listened to this morning on Spotify that is important. What is important is whether you listen to this song *every* morning, or how many other songs by the same artist you have listened to. The aggregate of these individual actions is what is important, as the pattern that emerges out of repetition and difference is what conditions predictability. At the same time, users' various clicks, 'likes', and sharing on Facebook inhibit other possible futures from happening.

Attention is thus used as a mechanism to both predict and inhibit the future, corresponding to what Lazzarato describes as the 'actualisation of power relations' through processes of integration and differentiation (2006: 174). The process of integration, following Lazzarato, explains how power relations gradually, step-by-step, become actualised through a logic of aggregation. This, too, is the goal of GraphRank, which seeks to calculate the trajectory of an object by aggregating bits of information into larger patterns or tendencies. On the other hand, as Lazzarato points out, 'the actualisation of power is not only integration, it is also differentiation: power relations are exercised to the extent that there is a difference between forces' (ibid.). This is the logic of GraphRank: as a direct response to user actions, it gives you more of what you have already paid attention to, at the expense of difference. This algorithmic logic is thus subject to a kind of performativity, as a 'formula that progressively discovers its world and a world that is put into motion by the formula describing it' (Callon, 2007: 320).

From public to personalised attention

In contrast to the discussions of whether Facebook is a public or private sphere, I suggest that Facebook in many respects installs a ‘community of those that in fact have nothing in common’ (Lingis, 1994). Although I use this expression in a literal sense to denote the ways in which the heavy personalisation of Facebook makes every user an island in and of himself, one could certainly make an argument about the sort of community of belonging that is stripped of any common sense of identity (a discussion that has been prominent within political philosophy for some time). In many ways, it makes sense to conceptualise Facebook not as a community founded on an idea of a same or shared identity, but rather as the relations formed across multiple identities that share only the fact that they have nothing in common (see Agamben, 1993; Esposito, 2010). Facebook prides itself on being the world’s largest social network – a community of friends – however, every user on Facebook is in fact separated from everyone else. No two News Feeds, profiles, or networks of friends are alike. As we have seen, EdgeRank and GraphRank are fundamentally geared towards generating a personalised view, tailored specifically to the ‘likes’ and tastes of individual users. These algorithms constitute important attention-selection mechanisms conditioned by user data and the ‘wisdom of friends’. Since liking web content has been rendered as content for News Feeds by the Open Graph protocol, friends’ endorsements have become an important factor in shaping what is considered ‘newsworthy’ in the context of Facebook’s algorithmically-curated News Feed.

The ways in which the software works as an algorithmic logic thus have a profound impact on the governance of visibility and invisibility in digital culture. However, the logic of this algorithmic curation of information on Facebook does not follow the principle of public attention associated with the function of media more typically (Webster, 2011). There is simply no ‘public’ to be addressed, because everything on Facebook is filtered in terms of the identity of specific users. This is evidenced by the ways in which EdgeRank changes what it shows to specific users on the News Feed based on their behaviour. As with any other information filtering and processing system, EdgeRank works by collapsing bits of information into comparable numbers to create calculable relations and differences.

However, there is a slight difference between the News Feed as a personalised attention economy, and other ‘algorithmic cultures’. Whereas a system such as

Amazon.com, for instance, is driven by the logic of *users like you* – designing user identity based on how many other people like you have paid attention to a certain book – the basis of Facebook is that nobody is like me. If everything in Facebook is tied to the user, the value attached to specific information and forms of interaction is specifically adapted to the needs, interests, and preferences of the individual. In anticipation of generating more activity and engagement on the platform, Facebook customises visibility by measuring and monitoring what are calculated to be meaningful relationships. Whether we are talking about the Open Graph as an infrastructure for providing ‘social context’ or the pervasiveness of the ‘Like’ button, every user action is inevitably ‘glued’ to and associated with friends’ tastes and likes. After all, as Facebook engineer Bosworth has suggested about the News Feed, it ‘knows who we keep an eye on and who we have cared for in the past’ (2007).

Conclusion: How software makes sense

In this chapter, I have argued that the capacity for attention in digital culture needs to be understood as a relational construct between users and their technical supports. Arguably, who or what is paying attention online, and to whom and with what effect is not easy to detect in an environment of automated decision-making agents. However, as the notion of governmentality implies, attention governs and is governed in concrete material contexts and assemblages. Protocols, algorithms, and buttons do not merely mediate modes of paying attention, but also shape the conditions of the sensible. The techniques and procedures used to direct users’ conduct and attention on Facebook involve assigning ranks to different nodes and edges; aggregating data into meaningful patterns; lowering the barrier for authentication systems in third-party apps to enable ‘frictionless sharing’; and the marking-up of external webpages so as to support the automated management of people and objects. An exploration of technicity thus highlights the ways in which attention emerges from a given constellation of technical elements and living bodies.

In this chapter, Facebook has been used to examine the ways in which attention is articulated in a medium-specific context. Deliberately disregarding the current focus on brain scans in media studies, this chapter has addressed some of the ways in which attention is organised around ‘technologies of government’. Rather than drawing upon data from brain scans to support an argument about the co-constitute nature of

attention, I have attempted to ‘scan’ the medium through a reading of the Facebook infrastructure. In so doing, my speculative scan focused on the development of the Open Graph protocol and ranking algorithms, demonstrating how attention is managed by Facebook to propagate a certain social order of continued participation. I have thereby suggested that there is a need to put greater emphasis on the ways in which attention is actually put to work, not just rhetorically, as part of popular media discourse about the media’s effects on our brains, but through an engagement with its technicity – how software ‘makes sense’,⁵⁵ how it is productive of new ways of attending to the world.

The protocological power explored in this chapter, however, is but one important infrastructural mechanism producing the condition for the sensible and intelligible on Facebook. Importantly, the Facebook experience is designed and delimited by its algorithmic architecture, particularly the EdgeRank algorithm, which is responsible for deciding what content to show users’ as part of the News Feed. In the next chapter therefore, I will focus explicitly on power of the EdgeRank algorithm, and the ways in which the algorithm can be said to construct a regime of visibility geared towards the pursuit of participation.⁵⁶

⁵⁵ ‘Makes sense’ in this context is meant to express the notion that software produces sensation and awareness, where ‘sense’ is used as a synonym for the senses/sensible and for the more affective meaning of attentiveness and awareness.

⁵⁶ Chapter 5 is based on a peer-reviewed article, published in a special issue of *Culture Machine* (2012) on ‘Paying Attention’, edited by Patrick Crogan and Sam Kinsley. I am grateful for the editors’ helpful comments on earlier versions of this chapter.

Chapter 6. EdgeRank algorithm: Becoming (in) visible

In the previous chapter I illustrated how the Facebook infrastructure, its protocols and algorithms, can be seen as a technology of government, geared towards shaping the conduct of its users. In order to unpick the ways in which users are governed and to what possible ends, I framed my discussion of the shaping power of communication infrastructures in relation to attention. Rather than trying to understand Facebook in terms of attention, my aim was to understand attention from the perspective of the software. By analysing the various techniques and procedures of the Facebook platform, specifically in terms of the Open Graph protocol, I was able to point to some of the ways in which attention is organised and managed by the platform. Contrary to for example television, with its focus on drawing collective or public attention towards certain events or situations (see Dayan, 2009), the Facebook infrastructure is geared towards the construction of personalised attention. The ways in which the coded systems of Facebook are set out to work, architected and managed, not merely turns attention into a data shadow of individual users but also enrolls users into active, or should we say passive, producers of attention. In this sense, I argued that attention, in the medium specific context of Facebook, should not merely be understood as a form of spectatorship but rather as a form of conduct. As such, the behaviour of users, their multiple actions of clicking, ‘liking’, commenting etc. are utilised by the platform as tokens of attention. The attention economy implemented by Facebook thus follows a data-driven logic, whereby the technical infrastructure, techniques and procedures guide the user to generate data in a certain way that subsequently provides the source for the formation of attention – always already tailored to each specific user.

In this chapter, I want to pick up on the important role that algorithms play in governing the conditions of the intelligible and sensible on Facebook. While users feed the software system with raw data, the techniques and procedures to make sense of it, to navigate, assemble, and make meaningful connections amongst individual pieces of data is increasingly being delegated to various forms of algorithms. If it is true that algorithms are increasingly taking over the logic of news editors and content managers on the Web, how can we begin to understand this kind of algorithmic intervention into what can be said to constitute one of the most important sites for

news today – Facebook’s News Feed? What are the principles and logics of this algorithmic form of ‘news’ editing? Whom or what does the algorithm governing News Feed ‘want’ us to pay attention to?

While attention was used as a way to address the driving force of the Facebook platform as an infrastructure in the previous chapter, visibility forms the main focus of this chapter. Indeed, one of the core functionalities of the media pertains to making something or someone visible. As Marshall McLuhan argued, media are precisely the ‘extensions of man’ because they aid humans to see and sense things that they otherwise could not. Sound for instance is made visible by the wings of an airplane when they break the sound barrier (McLuhan 1964: 24)⁵⁷.

In this sense, visibility becomes a medium-specific property, where different media have the capacity to produce different modalities of visibility. Of course it can be argued that making sound visible presumably was not the main intention of the airplane designers. But as we are reminded of, technologies have a politics. They are designed to do certain things, according to a specific logic. Sometimes the effects that technologies generate are intentional, and sometimes they are not. As I will show, although technologies are not deterministic in the sense that everything is calculated upfront, generating intentional consequences, they are heavily controlled and regulated. This, of course, is one of the key operating principles of the media industry, controlling who or what is made visible to whom, when.

The regime of visibility associated with Web 2.0 connects to the notion of empowerment, as social media are commonly thought of as having greatly expanded the chances of users and citizens to have their say, of being recognised as subjects with a voice. On the other hand, ubiquitous computing with increased deployment of surveillance technologies has often been associated with a sense of disempowerment. The increased surveillance potential of new media technologies are often depicted negatively. However, surveillance all too quickly gets equated with the capacity of technology to map, track and store biometric and affective data for the controlling of bodies in time and space. This chapter argues for the importance of revisiting the idea of the technical and architectural organisation of power as proposed in the writings of

⁵⁷ Remember that for McLuhan media should be understood quite broadly, so everything that somehow extends the human nervous system needs to be considered a medium.

Foucault, by highlighting an analytics of visibility, rather than merely transposing the concept of surveillance onto new objects. Becoming visible, or being granted visibility is a highly contested game of power in which the media play a crucial role.

While Foucault did not connect his theory of visibility specifically to the media, the framework he developed in *Discipline and Punish* (1977) helps illuminate the ways in which the media participate in configuring the visible as oscillating between what can and should be seen, and what should not and cannot be seen, between who can and cannot see whom. Examining new modalities of visibility thus becomes a question of *how* rather than *what* it is made visible, through which specific politics of arrangement, architecture and designs.

In this his chapter I investigate the notion of mediated and constructed visibility through a close reading of the News Feed and its underlying operational logic, the EdgeRank algorithm. I argue that Foucault's idea of an architectural constructed regime of visibility as exemplified in the figure of the *Panopticon* makes for a useful analytical and conceptual framework for understanding the ways in which the sensible is governed in social networking sites. The intention is not so much to offer a definite account of the role played by Facebook in capturing the world in code, but to open avenues for reflection on the new conditions through which visibility is constructed by algorithms online. Especially in a state of pervasive *visuality* we need to take a step back and ask not for the visual manifestations and representations of bodies in code, but of the actual configuration of such manifestations in and through the media, that is how visibility is constructed, through which technical measures?⁵⁸

To explore the ways in which algorithmic architectures dynamically constitute certain forms of social practice around the pursuit of visibility, the first section of this chapter gives an account of how previous research has conceptualised the issue of mediated visibilities. I then move to the case study, providing a detailed description of the known operational principles of the EdgeRank algorithm. The third section revisits the Foucauldian analytics of visibility, arguing that the algorithmically defined

⁵⁸ It seems that media studies has been mostly preoccupied with the issue of *visuality* understood as 'modes of expression', rather than visibility understood as something that goes beyond the merely visual, as something that designates a general mode of awareness about someone or something (Daniel Dayan's work is instructive in this regard).

visibility constructed through EdgeRank functions as a reversal of the regime instantiated by the Panopticon.

Media visibility

The media represent key mechanisms in the sorting, classification and ranking of the social field. In the context of the mass media, selecting and granting visibility has been described extensively using terms such as framing (Entman, 1993; Goffman, 1974), gatekeeping (Lewin, 1947) and agenda setting (McCombs and Shaw, 1972)⁵⁹. The media industry is built around parameters of visibility. Without wanting to illuminate, to expose or inform publics about something previously unknown, there is no apparent need for the media. As John B. Thompson points out,

Visibility is shaped by the distinctive properties of communication media, by a range of social and technical considerations (such as camera angles, editing processes and organizational interests and priorities) and by the new types of interaction that these media make possible (2005: 35-36)

Whereas print could be managed by political powers to control visibility in a desired way, the auditory quality of radio allowed for a distinct kind of intimacy. Later, television with the use of camera angles and close-up frames could render even the most detailed facial expressions and mannerisms of public personas (Thompson, 2005). Different media forms thus instigate different forms of visibility. The printing press is a carefully managed space of visibility organised through human editorial practices. Radio constructs a form of visibility that operates through sound rather than images, exemplifying the notion that visibility should not be confused with visuality.

Television, like film, exemplifies the various technical means of arranging and organising attention. It does so by constructing certain regimes of visibility through the deployment of angles, shots, frames, time and spatial techniques. While Thompson is more concerned with mediated visibility, in terms of its social repercussions on political life, his genealogy of different media forms and the kinds of

⁵⁹ One of the core news values indeed is about the showing or exposing something or someone. Investigative journalism is all about exposure and making previous invisible aspects come to light. Framing allows for showing in a specific way and thus act as representational devices. Frames are lenses through which interpretation and meaning-making can occur. In the context of mass media, the concept of frames has been widely used to describe the ways in which the media contribute in constructing social phenomena by encouraging certain ways of interpreting while silencing alternative interpretations.

visibility they instigate also support a view on visibility as medium specific. To use Marshall McLuhan's famous truism, the medium is the message, which is also to say that it is the medium that make the messages visible in the first place, governing visibility in a certain direction. Media as selection, sorting and framing mechanisms, ultimately point to the fact that media visibilities are never neutral; it is always about making the content meaningful.

With the advent of the Internet and the Web, the medium specificity of the architectural organisation of visibility, i.e. what can be seen and heard, to a large extent became a question of software. Many scholars have rightfully observed that the Internet has indeed contributed to new forms of visibilities; for instance, in facilitating the visibility of 'counter publics' exemplified in anti-globalization and the Zapatistas movements (Dahlberg, 2007) and the visibility of everyday citizenship (Bakardjieva, 2009). However, inquiries into mediated visibilities need to go beyond the visual and linguistic signification in accounting for ways of appearing online. Following a software sensitive approach usefully directs attention to the ways in which software functions as a sociotechnical actor capable of influencing users' practices and experiences on the Web. As David Beer has argued, Web 2.0 is fundamentally governed by the various sorting and filtering algorithms determining what the user encounters online (2009).

The chief actor in the digital ranking game of relevance is Google with its PageRank algorithm, developed by Sergey Brin and Larry Page in 1998. PageRank effectively ranks and shows websites based on an underlying assumption about relevance and importance. As Graham and Zook point out,

PageRank was modeled after academic citation literature as an objective means to measure the worth of a webpage, i.e. it assumes that the number of hyperlinks to a webpage provide some indication of the importance or quality of that page (2007:1323).

PageRank determines the importance of a webpage for a given search term based on the amount of incoming links by other websites and their perceived authority. While PageRank is still the major technological component used to 'give back exactly what the user wants', there are now over 200 different signals or factors that determine the

importance of a webpage (Google, 2012)⁶⁰. Importantly, PageRank and the other signals used to algorithmically determine importance are constantly changed and tweaked. Importance is by no means a static measure as what counts as important is dynamically defined in relation to the quality and amount of content coming online and indexed by Google. According to Google, they make roughly 500 changes to their search algorithm in a typical year (Google, 2011a). For example, one major algorithmic change in Google was the so-called ‘Panda’ update implemented in February 2011, where variables were changed to find and give more presence to ‘high-quality’ sites (see Google, 2011b). Panda was particularly noticeable as the update affected some 12 per cent of all Google queries. Moreover, in November 2011, another quite noticeable update was implemented, an algorithmic response to the proliferation of real-time data flows. Attempting to provide users with ‘fresher’ results, the algorithmic update resulted in 35 per cent of the searches being impacted (see Google 2011c).

These automated and calculable processes have largely influenced the ways in which people access information on the Web. Algorithms, such as the ones deployed by Google, fundamentally shape knowledge and meaning making practices online. It has become a truism that what does not show up on Google does not exist. The power to determine the logics through which information about a given topic gets on top of search results allows the algorithms to control and regulate the political economy of the Web. A combination of linguistic cues in the keywords of search strings along with patterns of query reformulations, relationships with html metadata tags, link structure and authoritativeness of the sites constitute key elements of the technical and social architecture of how search engines work (Granka 2010). As Pasquinelli (2009) notes, PageRank has become the most important source of visibility on the Web today. While Google and PageRank have received a lot of scholarly attention (see for instance Hagittai, 2007; Hellsten et al., 2006; Introna and Nissenbaum, 2000) the *other* powerful source of visibility on the Web today – EdgeRank – has to my knowledge hardly been critically scrutinised at all.

⁶⁰ As Google explains in their about section on technology: ‘Co-founder Larry Page once described the “perfect search engine” as something that “understands exactly what you mean and gives you back exactly what you want.” We can’t claim that Google delivers on that vision 100 per cent today, but we’re always working on new technologies aimed at bringing all of Google closer to that ideal’ (accessed March 12, 2012)

Studying algorithms

Despite society's increasing dependency on algorithms, they 'are rarely discussed in themselves and rarely attended to as objects of analysis' (Mackenzie, 2007:93). But what does it mean to critically scrutinise the EdgeRank algorithm? How to critique Google, Amazon, Facebook, or someone else's algorithms when they remain black-boxed and essentially 'wired shut' (Gillespie, 2007)? As I have already touched upon, I suggest that we may approach algorithms by means of what Bogost (2007) refers to a 'white-box analysis', by being attentive to the 'operational logic' of the algorithm. This implies a reading of the software with only a partial knowledge of the underlying code that produces the effects. Although we cannot always actually get under the bonnet of algorithms and computational processes, we might begin our investigations by asking the following: what are algorithms suggestive of? What kind of assumptions do they seem to embody? Which possibilities do they open and close? What kind of connections are being constructed, and how are these associations made? To what do they seem to draw attention? What are they supposed to assist users (both human and nonhuman) in accomplishing? What is their vision of relevance and importance? And finally, what do the algorithms set out to govern?

As the world is increasingly infused with algorithms, there is a need to better understand the politics they embody. Algorithms give the impression of neutrality, yet they are carefully crafted and put into play by engineers. As Tarleton Gillespie argues – using the Twitter trend algorithm as a case in point – there is often a tension between what we expect algorithms to be and what they in fact are (2011). For example, despite the fact that the Occupy Wall Street protests received so much media coverage and provoked so many Twitter discussions, why were not the widely-known hashtags #occupywallstreet and #occupyboston 'trending'? Importantly, 'the algorithms that define what is "trending" or what is "hot" or what is "most popular" are not simple measures, they are carefully designed to capture something that the site providers want to capture' (Gillespie, 2011). The connections, recommendations, suggestions, and associations made by algorithms need to be addressed by questioning why certain items are related to each other, and to what possible end. Such an endeavour is in line with with a Foucauldian-inspired diagrammatics of power. It is precisely because algorithms are associative devices, with the capacity to construct

certain connections at the expense of others, that there is a need to question the logics underlying their articulations.

Algorithms, then, embody and carry with them assumptions about the situations on which they are supposed to act. A software-sensitive approach to the question of mediated visibility thus needs to take into consideration the embodied assumptions and values of the algorithms at play. In PageRank, there are a number of assumptions built into the algorithm – about the nature of relevance, for instance. As we will see, EdgeRank too carries with it a number of assumptions about what constitutes relevancy, or ‘interestingness’, as Facebook calls it. Although displaying interesting content on users’ News Feeds remains the end goal, the steps that go into determining the path to this goal depends on a plethora of different factors. Unlike the Google maps example provided in Chapter 3, where measuring the distance between point A and B constitutes more or less stable variables (as geographical points do not change), interestingness in Facebook cannot be measured as easily.

Algorithmic visibility

News Feed and the EdgeRank algorithm

At the time of writing this, Facebook’s primary feature is the ‘News Feed’, forming part of the first page that users access when logging on to the site. As of August 2011, the ‘News Feed’ makes up the centre column of a user’s home page and represents a constantly updating list of stories from ‘friends’ and ‘Pages’ that a user has a relationship with on Facebook⁶¹. The News Feed is further divided into two different versions, the default ‘Top News’ and the ‘Most Recent’ feed. According to the Facebook help centre, the difference between the two is that ‘Top News aggregates the most interesting content that your friends are posting, while the Most Recent filter shows you all the actions your friends are making in real-time’ (Facebook 2011, emphasis added).

⁶¹ Facebook changed its interface and News Feed design on 20 September 2011. This chapter is based on a study of the News Feed *before* these changes occurred and refers to how the feeds worked and looked between April-August, 2011.


Akin to the algorithmic logic of search engines, Facebook deploys an automated and predetermined selection mechanism to establish relevancy (here conceptualised as most interesting), ultimately demarcating the field of visibility for that media space. As Kincaid (2010) explains, every item that shows up in your News Feed is considered an ‘Object’ (i.e. status update, uploaded picture). Every interaction with the Object, for instance through a ‘Like’ or a Comment, creates what Facebook calls an ‘Edge’. EdgeRank, the algorithmic editorial voice of Facebook, determines what is shown on users’ Top News by drawing on different factors relating to the Edges. At least three different components are key to determining the rank of an Edge:

(1) Affinity. This pertains to the nature of the relationship between the viewing user and the item’s creator. Here the amount and nature of the interaction between two users is measured. Sending a friend a private message or checking his or her profile on a frequent basis heightens the users’ affinity score to that particular friend.

(2) Weight. Each Edge is given a specific ‘weight’, depending on how popular or important Facebook considers it to be. Therefore, not every Edge gets weighted the same. Some types of interactions are considered more important than others. Arguably, a Comment has more importance than a Like.

(3) Time decay. Probably the most intuitive component relates to the recency or freshness of the Edge. Older Edges are considered less important than new ones.

EdgeRank is calculated based on the multiplication of the Affinity, Weight and Time Decay scores for each Edge (see figure 2). Becoming visible on the News Feed, appearing in that semi-public space, depends on a set of inscribed assumptions on what constitutes relevant or newsworthy stories. How many friends are commenting on a certain piece of content, who posted the content, and what type of content it is (e.g. photo, video, or status update) are just some of the factors at work in determining the rank of an Edge. The higher the rank, the more likely it will be that an Object appears in the user’s feed (Kincaid, 2010).

6. NFO: News Feed Optimization 
EdgeRank

$$\sum_{\text{edges } e} u_e w_e d_e$$

u - affinity score between viewing user and edge creator
w - weight for this edge type (create, comment, like, tag, etc.)
d - time decay factor based on how long ago the edge was created

Figure 2 EdgeRank formula

The algorithm is based on the assumption that users are not equally connected to their friends. Some friends thus ‘count more’ than others. The friends that count more are those with whom a user interacts with on a frequent basis, or on a more ‘intimate’ level; say by communicating with a friend via ‘Chat’ rather than on the ‘Wall’. This point becomes especially evident when considering the most recent changes to the operational logic of the News Feed. During February 2011, Facebook changed the settings and options for the ‘Most Recent’ feed. Two basic settings were implemented, a seemingly unfiltered one, showing stories from ‘All of your friends and pages’ as well as a filtered one displaying only ‘Friends and pages you interact with most’.

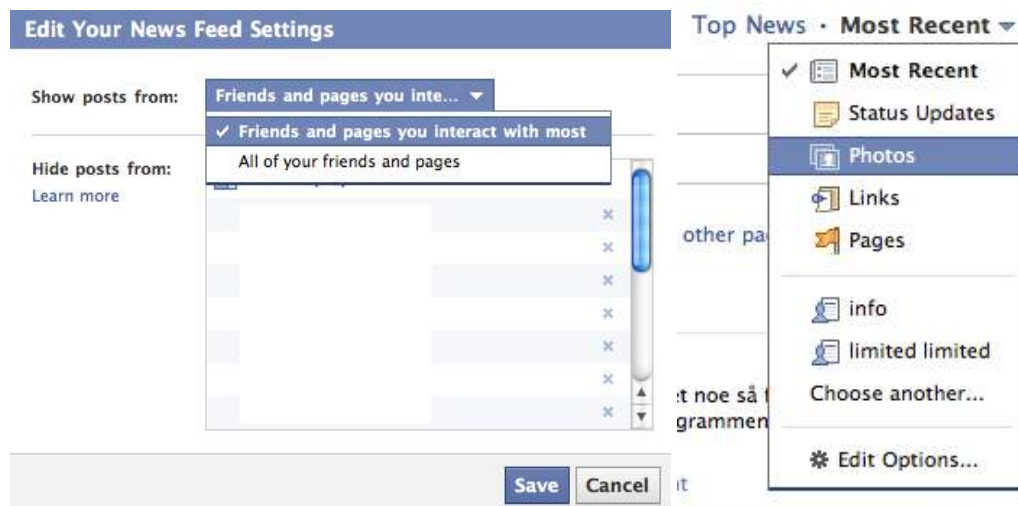


Figure 3 The most recent feed settings with default set to 'Friends and pages you interact with the most'. Screen short from February 25, 2011.

It was not so much the fact that Facebook changed the News Feed feature yet again that brought about much public outcry on the matter, but the fact that Facebook had changed the default setting of the Most Recent feed to 'Friends and pages you interact with most' so that the feed most users believed to represent *every* update from *every* friend in a real-time stream, in fact became an edited one, much like the Top News filter. Perhaps what caused the most controversy was the fact that this change in default occurred without Facebook notifying its users' about it. The option to change the default was tucked away at the bottom of a drop down menu next to the 'Most Recent' tab in the 'Edit Options' tab (see figure 3). This change in default caused a lot of content to be hidden away from users without their knowledge. Users who had noticed the change would ask about where all their friends had gone, warning others about the fact that the changes 'mean that you are not seeing everything that you should be seeing' (Hull, 2011). There are at least two interesting assumptions apparent in this. First, there is an idea about what should be visible. Second, there is a notion that Facebook acts ideologically in that the platform is hiding something from people's view. But what is it that *you should be able to see*? Clearly, there is a discrepancy between what users think they should be seeing and what Facebook thinks users should be seeing.

The EdgeRank algorithm is furthermore geared towards highlighting certain types of Edges while downgrading others, where the type of interaction becomes a decisive factor. Chatting with someone on 'Facebook Chat' presumably counts more than

‘liking’ someone’s post. There is a certain circular logic embedded in the algorithm. In order for you to like or comment on a friend’s photo or status update they have to be visible to you in the first place. Any time a user interacts with an Edge, it increases his or her affinity towards the Edge-creator. For instance, we can assume that comments outweigh ‘likes’ as they require more individual effort. The weight given to certain types of Edges, moreover, is likely to depend on the internal incentives that Facebook may have at any given point in time. If the objective for Facebook is to promote a certain product, for instance the ‘Questions’ feature or ‘Places’, interactions with these features will probably be ranked higher than others. This is understandable. News Feed is the best way to increase awareness of new (or neglected) features and functionalities.

The algorithm is not merely modelled on a set of pre-existing cultural assumptions but also on anticipated or future-oriented assumptions about valuable and profitable interactions that are ultimately geared towards commercial and monetary purposes. As I argued in the previous chapter, the anticipatory logic inherent in the functioning of these algorithms are not primarily geared towards confirming some pre-existing cultural logic but rather to model a mode of government that has the capacity to take into account the probability of subjects’ actions. By looking at the ways in which the specificities of the Facebook platform, exemplified here through the EdgeRank algorithm, enables and constrains ways of becoming visible online, we can begin to rethink regimes of visibility that hinge on and operate through algorithmic architectures. In doing so this chapter expands on Foucault’s idea of ‘panopticism’ as it provides a useful and still highly relevant analytics for understanding the ways in which visibility is technologically structured.

Rethinking regimes of visibility

Panopticism

Foucault operates with two basic notions of how things are made visible or shown, exemplified in his notion of the spectacle and surveillance. It is not just a matter of what is seen in a given historical context, but what can be seen and how the realm of the seeable and sayable is constructed in order to make a particular dispositif of visibility appear. As Thompson (2005: 39) explains, whereas the spectacle designates

a regime of visibility in which a small number of subjects are made visible to many, surveillance mechanisms, on the other hand, are connected from the 16th century onwards to the emergence of disciplinary societies, in which the visibility of the many is being assured by a small number of subjects. Surveillance as a mode of visibility was famously exemplified in the architectural arrangement of the Panopticon.

Adapting the figure of the Panopticon from Jeremy Bentham, Foucault sought to explain the regulatory force of power inherent in specific architectural compositions. The idea of the Panopticon designates an architectural vision of a prison, a circular building with an observation tower in the middle. The principle of this architectural vision is thus to render the subject, whether ‘a madman, a patient, a condemned man, a worker or a schoolboy’ (Foucault, 1977:200), in a state of permanent visibility. As Foucault explains: ‘All that is needed, then, is to place a supervisor in a central tower’ in order to create the impression that someone might be watching (ibid.). More so than actually placing a prison guard in the tower, the purpose of the architectural design is to create a space where one can never be certain whether one is being watched or not. As Foucault elaborates:

Hence the major effect of the Panopticon: to induce in the inmate a state of conscious and permanent visibility that assures the automatic functioning of power. So to arrange things that the surveillance is permanent in its effects, even if it is discontinuous in its action; that the perfection of power should tend to render its actual exercise unnecessary; that this architectural apparatus should be a machine for creating and sustaining a power relation independent of the person who exercises it; in short, that the inmates should be caught up in a power situation of which they are themselves the bearers (1977:201).

The uncertainty associated with the possibility of always being watched, inevitably leads the subject to adjust his or her behaviour accordingly, so as to behave as if they indeed would be permanently observed. Surveillance thus signifies a state of permanent visibility. The novelty of Foucault’s notion of visibility, constructed by the specificities of the historical contingent architectural apparatus, lies precisely in highlighting the technical organisation of power. As Foucault points out, the Panopticon is not a dream building: ‘it is the diagram of a mechanism of power reduced to its ideal form [...] it is in fact a figure of political technology’ (Foucault, 1977:205). Power, we may recall, ‘has its principle not so much in a person as in a certain concerted distribution of bodies, surfaces, lights, gazes; in an arrangement whose internal mechanisms produce the relation in which individuals are caught up’

(Foucault, 1977:202). By highlighting the diagrammatic function of panoptic surveillance, Foucault provides a forceful analytical framework for understanding different modalities of visibility and the mechanisms by which it is being arranged.

As Rajchman points out in his discussion of Foucault: ‘Architecture helps ”visualize” power in other ways than simply manifesting it. It is not simply a matter of what a building shows ”symbolically” or “semiotically”, but also of what it makes visible about us and within us’ (1988: 103). Conceiving of visibility as an organisation of power in both a negative and a positive sense, Foucault shows that ‘spaces are designed to make things seeable, and seeable in a specific way’ (Rajchman, 1988). Prisons, hospitals and social networking sites are essentially spaces of ‘constructed visibility’. The realm of visibility created by the panoptic architecture did not work primarily through a certain iconography, nor a visual semiotic regime, but first and foremost through the technical structuring of a way of being, implementing an awareness or attentiveness to the constant possibility of inspection. To highlight visibility as a system, a diagram, is to highlight the ‘distribution of individuals in relation to one another, of hierarchical organization, of dispositions of centres and channels of power’ (Foucault, 1977:205). It is precisely this notion of a material or technical structuring of visibility that seems especially interesting and relevant in terms of new media. The spaces designed by the (im)material conditions of the software are similarly designed to make things visible, and thus knowable in a specific way.

Threat of invisibility

The mode of visibility at play in Facebook, as exemplified by the News Feed and its EdgeRank algorithm differs from that of disciplinary societies in one particularly interesting way. The technical architecture of the Panopticon makes sure that the uncertainty felt by the threat of permanent visibility is inscribed into the subject, who subsequently adjusts its behaviours. While one of the premises of the panoptic diagram pertains to even distribution of visibility, in which each individual is subjected to the same level of possible inspection, the News Feed does not treat individuals equally. There is no perceivable centralised inspector that monitors and casts everybody under the same permanent gaze. In Facebook there is not so much a ‘threat of visibility’ as there is a ‘threat of invisibility’ that seems to govern the

actions of its subjects. The problem is not the possibility of constantly being observed, but the possibility of constantly disappearing, of not being considered important enough. In order to appear, to become visible, one needs to follow a certain platform logic embedded in the architecture of Facebook.

There is now a whole industry being built around so-called 'News Feed Optimisation' akin to the more established variant, search engine optimisation. Marketers, media strategists, PR firms all have advice on how to boost a brand's visibility on Facebook. A search for 'EdgeRank' on Google reveals that the first twenty returns are almost entirely by social networking marketing or otherwise e-commerce related businesses. While many individual users may not be aware of the algorithmic politics behind the News Feed, this has become one of the main concerns for businesses and organisations that want to reach their desired audience. According a market report: 'Making it to that News Feed is crucial. No matter how interesting a brand's content program, visibility is required to get users to interact with it' (Shahani et al., 2011: 2). Similarly, Taylor suggests that: 'Facebook's EdgeRank holds all power of visibility' (2011).

The threat of invisibility should be understood both literally and symbolically. Whereas the architectural form of the Panopticon installs a regime of visibility whereby 'one is totally seen, without ever seeing' (Foucault, 1977:202), the algorithmic arrangements in Facebook install visibility in a much more unstable fashion: one is never totally seen or particularly deprived of a seeing capacity. As is the case in the Panopticon, the individual Facebook users can be said to occupy equally confined spaces. Like the carefully and equally designed prison cells, the user profile represents a schemata that 'provide fixed positions and permit circulation' (Foucault, 1977:148). Just as with the concrete machines (i.e. military, prisons, hospitals) described by Foucault, it is not the actual individual that counts in Facebook. This is why spaces are designed in such a way as to make individuals interchangeable. The generic template structure of Facebook's user profiles provide not so much a space for specific individuals but a space that makes the structured organisation of individuals' data easier and more manageable. The system, then, does not particularly care for the individual user as much as it thrives on the decomposition and recomposition of the data that they provide. However, whereas the architecture of the Panopticon makes all inmates equally subject to permanent visibility, EdgeRank

does not treat subjects equally, it prioritises some above others. Whereas visibility, as a consequence of the panoptic arrangement, is abundant and experienced more like a threat imposed from outside powers, visibility in the Facebook system arguably works the opposite way. The algorithmic architecture of EdgeRank does not automatically impose visibility on all subjects. Visibility is not something ubiquitous, but rather something scarce.

In order to see how many of the Most Recent posts actually made it to the Top News I conducted an experiment of what could be considered a process of ‘reversed engineering’ (see Chapter 4). Over the course of several months (March-September, 2011) I used my own personal Facebook profile to compare the contents of the Top News to that of the Most Recent. The most intensive investigation took place during April, 2011 where I did the comparison a couple of times a week and took screen shots of the entire Top News feed and manually counted the posts in the Most Recent feed. I took the oldest story published in the Top News and compared it to the amount of stories published in the Most Recent feed up to the same time stamp.

On a randomly selected day in April 2011, this amounted to 280 stories/updates published on the Most Recent feed, as opposed to 45 posts appearing in the Top News feed within the same timeframe. At first glance, only 16 per cent of the possible stories seem to have made it to the Top News. As time decay is one of the three known factors of EdgeRank, it is safe to assume that there is a higher probability of making it into the Top News the closer to real-time the story is published. In fact, my experiment showed that if a story/update was published within the last three hours, there was between 40 to 50 per cent chance of getting onto the Top News. In addition to selecting from the total amount of updates generated by friends from the real-time stream, Top News also displays its own tailored news stories that are *not* displayed in the Most Recent feed. These stories I call communication stories, as they construct a story out of two friends’ recent communicative interaction (see figure 4).



Figure 4 Communication stories. Screen shot from September 20, 2011.

Communication stories in Facebook typically take the form of ‘X commented on Y’s photo’ or ‘X likes Y’s link’. Taking these tailored stories into the equation, a better estimate for the 45/280 ratio would be a mere 12 per cent chance of getting in the Top News. No matter how meticulous the counting and comparing between the two feeds, the exact percentage of stories making it to the Top News remains somewhat obscure. Like Google’s PageRank the exact workings and logics of EdgeRank includes more factors than what is publicly known. While Affinity, Weight and Time Decay are key components of the algorithm structuring the regime of visibility in News Feed, it is safe to assume that other factors will affect the ranking and selection of Edges as well. What becomes apparent is that algorithms, especially those working at the heart of ‘data driven’ companies like Facebook and Google, occupy a peculiar epistemological position where some components are known while others are necessarily obscured.

Algorithms are fundamentally relational, in the sense that they depend on some kind of external input (data) in order to function. Algorithms do not just represent a rigid, pre-programmed structure, understood as ‘recipes or sets of steps expressed in

flowcharts, code or pseudocode' (Mackenzie, 2006: 43). They are also fluid, adaptable and mutable. This means that EdgeRank is not something that merely acts upon users' from above, but rather that power arises from its interrelationships with users. How EdgeRank will process the data that I provide, therefore, fundamentally also depends on *me*, as well as on my relationships with my 'friends'. For instance, Top News dynamically updates depending on how many times I visit Facebook, which makes it difficult to make a general claim about the percentage of stories making it to the Top News. On a random day in September 2011 I compared the same Top News feed *before* and *after* checking the Most Recent feed, which at that time counted over 300 new posts⁶². What I found was a change of about 34 per cent in stories displayed between the two instances of checking Top News⁶³. After having checked the Most Recent feed then, 16 of the 47 posts in the Top News feed had immediately changed.

The quite noticeable change in stories displayed can be explained by referring back to the workings of EdgeRank. In the first round of checking my Top News, EdgeRank seemed to put more emphasis on the 'time decay' mechanism knowing that it had been a while since I had last logged in to Facebook. After checking the Most Recent feed however, Facebook 'knew' I was 'up to date' again, replacing 16 of the previous posts with ones that EdgeRank had calculated to apparently be of greater 'interest' to me. All the 16 new stories displayed in the second round of checking had been posted between 12-23 hours from the time of checking and were either of the 'communication story' type or stories that had generated several Likes and Comments by others. While the algorithmic architecture works dynamically, thereby making analysis of its workings difficult, we can treat them as a particular way of framing the environments they work upon (Mackenzie, 2007). We may thus say that EdgeRank, acting as a gatekeeper of user-generated content, demarcates visibility as something that cannot be taken for granted. The uncertainty connected to the level of visibility

⁶² The Most recent feed used to have a counter next to it indicating the amount of new stories that had been posted and published by a user's Facebook connections since the last time they had checked the feed. The counter only went as far as 300 new posts, so any amount above the limit would just be indicated as '+300'. Usually, my most recent feed would reach this limit only after one or two days of not checking.

⁶³ In September 2011 I did this kind of comparison between my Top News feed before and after checking the Most Recent a couple of times and every time there was a considerable change in stories displayed between the first and second time of checking.

and constant possibility of ‘disappearing’ in relation to the ‘variable ontology’ of software, frames visibility as something quite exclusive. Becoming visible on the default News Feed is thus constructed as something to aspire to, rather than feel threatened by.

Visibility as a reward for interaction

In Facebook becoming visible is to be selected by the algorithm. Inscribed into the algorithmic logic of the default News Feed is the idea that visibility functions as a reward, rather than as punishment, as is the case with Foucault’s notion of panopticism. A different Top News sample from the experiment reveals the following: of 42 posts displayed, only three of the stories published by my ‘friends’ came without any form of interaction by others (that is, without any Likes or Comments). Eleven stories published were by pages that I had liked, none of which had generated more than one ‘like’. Out of the remaining friend stories, 15 were status updates with comments and/or ‘likes’, and 10 were of the tailored type where friends had either commented or ‘liked’ someone’s uploaded photo, video or shared link. My Top News was filled with stories that obviously signify engagement and interaction. Although distribution of the specific types of stories published varied over the course of the two months, stories without significant interaction seemed to be filtered out. The fact that there were almost no stories by friends that prevail on the Top News without any form of engagement by others strengthens the impression about the algorithmic bias towards making those stories that signify engagement more visible than those that do not.

As already mentioned, Top News displays its own tailored stories that do not appear in the Most Recent feed. The months’ worth of tracking my Top News showed a significant favouring of the form ‘X commented on Y’s photo’ followed by ‘X likes Y’s photo’. This suggests that photos are an important currency for getting on the Top News, as are friends’ interactions with these photos. In fact, Facebook is now the largest photo-sharing site on the Web, which it arguably would not have been without making interactions with ‘friends’ photos a particularly salient and visible form of social activity. Most of these types of stories are characterised by having many others, friends or friends of friends, also commenting or ‘liking’ the post. Of all the 45 stories published on my Top News another day in April 2011, 17 were communication

stories. What most of these 17 communication stories had in common was a high degree of interaction. For example, a typical story would say: ‘Anna commented on Claire’s photo’ along with ‘11 people like this’ and ‘View all 14 comments’. Not only does Facebook tailor specific stories for the Top News feed, these stories also receive a significant amount of visibility as opposed to other types of Edges. On average, communication stories made up a third of my entire Top News, with variations between 24 and 40 per cent. Since, on average, a third of the Top News stories do not appear in the Most Recent feed, the amount of stories making it into the Top News are even less than originally thought. While there is a higher chance of getting into the Top News if the post was published within the past three hours, continuously interacting on Facebook seems to be a better bet for becoming visible.

Participatory subjectivity

The threat of invisibility on Facebook, then, is not merely a symbolical phenomenon, but also literally quite real. While the regime of visibility created by Facebook may differ from the one Foucault described in terms of surveillance, understood as imposing a state of permanent visibility, discipline is still part of the new diagrammatic mechanisms. While it has become commonplace to argue for the transition of a disciplinary society into a control society after the post-industrial fact described by Deleuze (1992), I do not see a necessary contradiction between the disciplinary diagram and software-mediated spaces. Discipline simply refers to a diagram that operates by making the subject the ‘principle of (its) own subjection’ (Foucault, 1977:203). Discipline denotes a type of power that economizes its functioning by making subjects responsible for their own behaviour. As such, ‘discipline ‘makes’ individuals; it is the specific technique of a power that regards individuals both as objects and as instruments of its exercise’ (Foucault, 1977:170). It imposes a particular conduct on a particular human multiplicity (Deleuze, 2006: 29). It is important here to highlight that Foucault developed the notion of disciplinary power in order to account for the duality of power and subjectivation – effectuated by ‘training’ subjects to think and behave in certain ways and thus to become the principle of their own regulation of conduct. Through the means of correct training, subjects are governed so as to reach their full potentiality as useful individuals (Foucault, 1977:212). Foucault identified three techniques of correct training;

hierarchical observation, normalising judgement and the examination. In this sense we could say that EdgeRank exercises a form of disciplinary power as discipline in Foucault's words, 'fixes; it arrests or regulates movements; it clears up confusion; it dissipates compact groupings of individuals wandering about the country in unpredictable ways; it establishes calculated distributions' (1977: 219).

For Facebook, a useful individual is the one who participates, communicates and interacts. The participatory subject evidently produced by the algorithmic mechanisms in Facebook follows a similar logic to those techniques of correct training at work in sustaining disciplinary power. First, the very real possibility of becoming obsolete inscribed by the 'threat of invisibility' arguably constitutes a desire to participate. Here we can see the double logic inherent in Foucault's understanding of power, as both constraining and enabling. While visibility is constrained by the failure to conform to the inherent logics of participation, visibility is also produced and enabled by the same logic. As Foucault asserts: 'What is specific to the disciplinary penalty is non-observance, that which does not measure up to the rule, that departs from it' (1977: 178). Not conforming to the rules set out by the architectural program is thus punishable. That is, not participating on Facebook will get you punished by making you invisible.

Secondly, making it appear as if everybody is participating and communicating by emphasising those stories that generate a lot of comments and likes provides an incentive to like or comment as well. Simulation creates an impression, and it is precisely the power of impressions that Foucault thought was a driving force in the governing of the self. As Hoffman elaborating on Foucault's notion of disciplinary power explains: 'Disciplinary power judges according to the norm. He depicts the norm as a standard of behaviour that allows for the measurement of forms of behaviour as "normal" or "abnormal"' (2011: 32). By creating the impression that everybody participates, Facebook simultaneously suggests that participation is the norm. Normalisation, according to Foucault, created a 'whole range of degrees of normality indicating membership of a homogeneous social body but also playing a part in classification, hierarchization and the distribution of rank' (1977:184). EdgeRank, by functioning as a disciplinary technique, create subjects that endlessly modify their behaviour to approximate the normal. Because interaction functions as a

measure for *interestingness*, practices of liking, commenting and participation become processes through which the subject may approximate this desired normality.

Thirdly, the participatory subject created by the algorithm hinges on an underlying idea of popularity. Displaying Edges with a high degree of interaction clearly remediates some well-known cultural assumptions and mass media logics – popularity fosters further popularity. There is thus a circular logic to the way in which visibility is organised on Facebook. Being popular enhances the probability of becoming visible and thus increasing the probability of generating even more interaction. Being confronted with communication stories that encourage the user to ‘view all 14 comments’, ‘view all 9 comments’, and acknowledge that ‘Christina and 7 others like this’ enhances the impression that visibility is granted to the popular. EdgeRank by emphasising the perceived popularity of Edges, also reinforces a regime of visibility that runs counter to much of the celebratory Web 2.0 discourse focusing on democratisation and empowerment. While Facebook is certainly a space that allows for participation, the software suggests that some forms of participation are more desirable than others, where desirability can be mapped in the specific mechanisms of visibility, as I have suggested throughout this chapter.

Conclusion

Given the 800 million Facebook users who get affected by the ways in which algorithmic automated processes decide whether their content deserves to get on the ‘top’, it is surprising how little attention the infrastructures of Web 2.0 have received by media scholars at large. Taking up David Beer’s call for the need to explore and describe ‘power through the algorithm’ (2009: 999), I have examined Facebook’s EdgeRank as a form of disciplinary diagram, ultimately engaged in the material structuring of visibility. Drawing upon Foucault’s architectural framework as a way to analyse the ways in which the Facebook space is ‘designed to make things seeable, and seeable in a specific’ way (Rajchman, 1988), this chapter has argued that mediated spaces are never neutral with regards to the construction of specific regimes of visibility. Space, as Kitchin and Dodge argue, is constantly brought into being as an incomplete solution to an ongoing relational problem (2011:71). This is also reminiscent of algorithms as ‘proposed solutions to problems’ (Mackenzie 2006:46). Algorithms, like space, are ontogenetic – in becoming - simply because the problems

that require a solution continuously change. While the problem of showing the most ‘interesting’ content remains, what constitutes interestingness depends on the given context, as I have shown in my discussion of EdgeRank. This ontogenetic nature or the ‘variable ontology’ of software and algorithms constitute an argument for an ongoing research commitment to platforms like Facebook and its sociotechnical components. Facebook is never finished. As scholars we need to be vary and attentive to the ways in which our research objects change in an ongoing fashion. Many of the characteristics associated with disciplinary power described by Foucault, such as the function of enclosure, creation of self-control and the training of human multiplicity, are apt characterisations of the kind of enclosed architecture of Facebook and the subtle demands for participation and interaction. However if we follow Foucault in his understanding of surveillance as a form of ‘permanent visibility’, then, this notion fails to capture the algorithmic logic of creating modalities of visibility that are not permanent but temporary, not equally imposed on everyone and oscillating between appearing and disappearing.

While it is true that Foucault described organisations of power within a rather fixed technological and architectural form, the *idea* that architectural plans structurally impose visibility does not seem to come in conflict with the unstable and changing arrangements characteristic of new media. On the contrary, with reference to Foucault’s concept of panopticism, the aim of this chapter has been to argue for the usefulness of applying an analytics of visibility to (im)material architectures. Following Foucault’s assertion that ‘the Panopticon must be understood as a generalizable model of functioning; a way of defining power in terms of the everyday life of men’ (1977:205), I think that a diagrammatic understanding EdgeRank provides a constructive entry point for investigating how different regimes of visibilities materialise.⁶⁴

⁶⁴ An earlier version of Chapter 6 has been published as ‘Want to be on the top? Algorithmic power and the threat of invisibility on Facebook’ (2012), in *New Media & Society*, 14 (7): 1164–1180.

Chapter 7. Managed relationships: Assembling friendship

In this chapter I will continue my investigations into programmed sociality by including and expanding on the kinds of actors involved in producing new forms of sociality. Whereas the previous chapter concentrated exclusively on the power articulated through EdgeRank, this chapter takes a broader range of software actors into account to explore the powerful subjectifying effects that software gives rise to. I showed how the regime of visibility established on Facebook hinges on a form of algorithmic power that meter out which information is highlighted and overlooked, which associations are made or unmade, which resources are ranked high or low. I argued that EdgeRank contributes to the production of a form of participatory subjectivity that is subject to a compulsion to popularity generated by the algorithm. Participation on Facebook, I argued, can be understood in terms of the fear of becoming obsolete, what I called the ‘threat of invisibility’. This chapter seeks to further explore the contexts, workings, and implications of algorithms and other software processes by way of critically investigating the dynamics of ‘friending’, of self-other relations and the collective associations formed by human and non-human entities. In doing so, this chapter outlines the ways in which Facebook is organised and structured around sociality, which the platform conceptualises as ‘friendship’. Not only is friendship the name given to the social connections between users on Facebook, the individual subject is fundamentally addressed and positioned as friend.

The immense popularity of Facebook in recent years has been echoed by much academic research investigating issues of friendship on social networking sites (see for instance Lewis and West, 2009; Tang, 2010; Vallor, 2011). Most research, however, remains user-focused, investigating friendship as a pre-existing social category transposed into the realm of social networking sites. Yet, the configuration of friendship online is fundamentally technologically-driven and commercially motivated. Investigating the meaning of friendship in social networking sites therefore needs to include a critical understanding of the various technocultural processes (Langlois, forthcoming) shaping sociality online. Everything from setting up a profile on Facebook, connecting with other users and maintaining a network of friends requires an intimate relation with the software platform itself. Algorithms assist users in finding friends, supposedly ‘know’ user preferences and habits so that ones

‘important’ friends can be made more visible, help users ‘remember’ people from the past and prompt users to take certain communicative and relational actions. As van Dijck points out, ‘what is important to understand about social network sites is how they activate relational impulses’ (2012: 161). Not only is it important to understand *that* relationships are activated online, but also *how* they are being activated. By whom, for what purpose, according to what mechanisms? Understanding the contours of close social relationships is not just important from a general sociological point of view. How users in engaging with social relations online simultaneously closely interact with the machine, importantly illuminates the sociotechnical dynamics of programmed sociality.

Social networking sites are essentially designed and programmable spaces that encourage the user to carry out specific actions. This chapter addresses the sociotechnical ways in which Facebook *wants* friendships to be activated. The aim is to examine the specific mode of friendship produced in and through the Facebook ‘assemblage’. Taking the position that technology is not neutral, I argue that one must look at the specific ways in which sociality is wrapped in code, processed by algorithms, and afforded by various software features and functionalities. Here I specifically draw on perspectives from ANT and assemblage theory (see Chapter 2) that open up a space for seeing the range of actors involved in shaping sociality online, as well as the various processes and materialities at play in configuring how users are made to relate to themselves and others as friends.

Taking Facebook as a case study, this chapter focuses specifically on the role played by different software actors in the configurations of friendship online. As Facebook on a material level is software, its structural conditioning of friendship can be understood by paying attention to the materiality of cultural expressions that are embedded within various software processes, algorithms, and protocols. Part of the argument thus involves questioning the traditional view of friendship as something that occurs between two human beings. Contrary to the notion that ‘friendship clearly exists as a relation between individuals’ (Webb, 2003: 138), what I want to argue is that friendship on Facebook exists as a relation between *multiple actors*, not only human individuals, importantly involving algorithmic and commercial relations of forces. As already implied, users do not only forge connections with ‘friends’ via online platforms; the platforms themselves also contribute to the creation of these

social connections. Thus, as I will show in an examination of the Facebook platform, a plethora of actors, including non-humans, compete in defining the nature and meaning of friendship today: what it is, can, and should be, for whom and between whom. The question then, is how we understand the form of sociality called friendship, if we allow for a broader range of actors to play a decisive role in defining and forming these relationships? As we will see, developing an understanding of the sociotechnical dimension of friendship necessarily also entails paying attention to the apparent disconnect between established definitions of friendship and Facebook ‘friending’.

However, rather than engaging with the existing polemics of whether ‘Facebook friends’ is good and bad, or whether we are dealing with *real* or *unreal* friendship (see Briggles, 2008; Cocking and Matthews, 2001), this chapter asks for the *specific* mode of friendship constituted and reinforced by the software platform that is Facebook. I propose the concept of ‘algorithmic friendship’ as a way of understanding the ways in which algorithms and software have become active participants in our networked lives and information ecosystems, forming the ways in which users are made to relate to self and others. This chapter is structured as follows: first I briefly revisit some of the existing research on friendship online, before commenting on the kind of analytical perspective used to understanding the difference that non-human actors play in configuring friendship on Facebook. Although much of this theoretical framework was detailed in chapter 2, I think it is useful to remind the reader of some of the important insights from actor-network theory and assemblage theory in order to help frame the subsequent analysis and discussion of friendship.

The main part of the chapter looks at the different elements of friendship on Facebook by focusing on the role played by software actors in processes of initiating, maintaining, and performing friendship. The purpose is to show how sociality online needs to be understood on its own technocultural terms, meaning that social relations online cannot merely be understood from an ‘offline’ perspective. By being attentive to the intervention of algorithms and various software features we can begin to see how sociality online is not something that exists outside of software, but rather how software always already participates in its configuration.

Revisiting friendship online

There have been numerous accounts during the last decade on the nature of digital connections, interpersonal relations and the quality of life online. This emphasis on the social nature of media use is not at all surprising given the fact that friendship in many ways constitutes the guts of social media, so to speak. That is, the very concept of social media implies the production and performance of friendship. Issues that have been widely addressed within new media research on friendship online include: the qualitative differences between offline and online friendships (Buote et al., 2009; Parks and Roberts, 1998), the meaning of friendship in social networking sites (boyd, 2006; Lewis and West, 2009), impression management and interpersonal communication (Tong et al., 2008; Lüders, 2009) and social capital (Ellison et al., 2007). Internet research prior to social media (before 2004 roughly) fostered much research on the quality of online friendships, producing statements such as, ‘Though we think internet “friendship” is quite inferior to non-virtual friendship, we do not think that it is necessarily bad in itself, and indeed for some people it clearly provides an important good’ (Cocking and Matthews, 2001:224).

With the rise of social networking sites, the rather gloomy accounts of inferior online friendships set against the more truthful and genuine ‘real-life’ friendships lost some of its grips, as research showed that what had been taken as a radical disconnect in fact complemented each other. Especially the scholarship of danah boyd and her ethnographic work on teenagers and social networking sites, showed how most online friendships have an offline counterpart, meaning that people tend to become friends with people online with whom they are already have an offline connection with (as opposed to establish new friendships) (boyd 2006). As such, ‘online and offline are not separate worlds—they are simply different settings in which to gather with friends and peers’ (boyd, 2010). Boyd also makes a distinction between ‘friends’ and ‘Friends’, where the latter refers to the more loosely defined friendships on social networking sites. Because these sites conflate the range of possible social relations to the category of friend only, users end up having hundreds of ‘Friends’ without necessarily adopting the traditional meaning of the term. According to boyd, users simply override the term ‘Friend’ to make room for a variety of different relationships (2006:4).

Social media and especially Facebook have altered the ways in which friendships are

performed and the practices surrounding friendship. As documented by boyd (2010), new friendship-driven practices are not just shaped by the social, cultural, and economic conditions that surround people, but are inherently configured by the technology supporting and underlying these practices. Take the concept of ‘friend lists’ for instance. As boyd points out,

One of the ways in which social media alter friendship practices is through the forced—and often public—articulation of social connections. From instant messaging “buddy lists” to the public listing of “Friends” on social network sites, teens are regularly forced to list their connections as part of social media participation. The dynamics surrounding this can directly affect friendship practices (2010:93-94)

Boyd describes the important, but often overlooked, interconnection between software features and the culture and social practices that surrounds it. As many writers of software studies have pointed out, analysis of contemporary culture needs to pay more attention to the various software features and functionalities and the kinds of practices that the software afford in understanding how culture works in software society (Lovink, 2007; Manovich, 2011). While the introduction of ‘friend lists’ as an organising feature of social networking sites may seem banal at first sight, the ways in which such technical affordances articulate new modes of sociality needs to be questioned. What is implied by the fact that millions of young people all of a sudden become accustomed to sorting their social connections in publicly displayed lists? Why do these lists automatically come with counters that keep track of the number of friends? And why does the Facebook user not have a choice to turn off the number counter as part of the software settings? The kind of sociality enabled and produced by social networking sites arguably needs to be critically scrutinised along these lines if we are to make sense of these new kind of technocultural practices.

It can be argued that young people growing up these days barely remember a time without the concept of sending out so-called ‘friend requests’, ‘poking’ or ‘linking’ their friends status updates on Facebook. What we need to keep in mind, though, is the arbitrariness and constructedness of such functionalities and features. Not very long ago these software functionalities did not exist, now they are part of the cultural mainstream. As Lovink reminds us, the more people are online, ‘the more important it is to understand that the technical architecture of the tools we use is shaping our social experiences’ (2007: 214). Boyd (2010) uses the example of the MySpace feature ‘Top

Friends' to highlight the often quite curious and forced ways of governing usage practices, in this case to decide and showcase who their actual close friends are. While MySpace might have designed the feature to make it easier for users to keep track of their friends, the meaning and implications it has for users might become quite the opposite, complicating matters and turning these online spaces into sites of 'social drama'. As boyd points out, 'the Top Friends feature is a good example of how structural aspects of software can force articulations that do not map well to how offline social behavior works' (2010:103).

While boyd importantly puts the power of software on the social media research agenda, I am not aware of any studies that have tried to understand the production of friendship by focusing on the structural aspects of software and boyd's focus remains largely focused on the ethnographic and user side of things. In this chapter I therefore want to take up the challenge of trying to understand friendship from a software studies perspective. Rather than drawing on users self-accounts like boyd, this chapter seeks to question the kind of sociality structured by software by way of comparing it to traditional held conceptions of friendship.

Assemblage as an analytical framework

In order to critically investigate the nature of friendship on Facebook, one must also investigate Facebook itself. Facebook is not just a simple website or social networking platform, a blank canvas upon which sociality is allowed to take place. Rather, as website and platform, Facebook constitutes an assemblage of various relations and actors, including people, technology, software processes, social practices, and values. This view of the Facebook platform as a sociotechnical entity, or assemblage, resulting from heterogeneous relations between a diverse set of actors, is one that builds on a relational account of the human-technology dyad. The concept of assemblage usefully points towards the ways in which reality and its specific entities are, above all, compositions of diverse elements that when put together have the capacity to act (see Deleuze and Guattari, 1987). The concept of agency underlying the analytical framework surrounding friendship used here echoes ANT (see Latour, 2005), in that everyone and everything can be an actor, as long as the action influences or provokes an action by someone or something else. The critical approach employed does not attempt to debunk previously held conceptions of

friendship online, but rather examines the ways in which friendship is being assembled in specific ways.

For my purposes, framing Facebook as an assemblage helps viewing the platform as a ‘process, an ongoing organizing of multiplicities, of relations between elements and forces, that produces affects’ (Coonfield, 2006: 290). This is to say that Facebook, by organising heterogeneous relations in a specific way, constitutes a productive force: it makes new relations possible. The concern is not so much with what the assemblage is, but rather with what it can do, and what it is capable of.

Here I draw on Foucault’s conceptualisation of power as productive, and particularly the notion that power is productive of certain ways of becoming a subject (Foucault, 1982). In rather subtle ways, buried underneath the signifying surfaces of the computer interface, embodied in abstract protocols, written in calculable documents and wired materials, software engages in processes of subjectivation. Seen this way, Facebook constitutes a platform for the production of ‘computer-aided subjectivity’ (Guattari, 1989: 133). With regards to new media, a critical understanding of the production of subjectivity as fundamentally intertwined in technical and institutional mechanisms (Foucault, 1982; Guattari, 1996: 197) provides a necessary framework for considering programmed sociality in and through social networking sites. Seeing friendship as an assemblage provides a lens through which the power of software to produce new modalities of subjectivation can be analysed. Thus, what is of importance here is to be open to the diverse range of actors and relations that come together to make something called friendship meaningful. How does the ‘machine’ operate to produce a distinct form of friendship? That is, how is friendship articulated within the software-subject assemblage of Facebook?

Software-generated friendship

This chapter focuses on exploring the relationships between software and users, that is, on questioning how the friend as a social category with a specific field of cultural values, norms and practices is configured in and through software. In the following, I will explore some of the features and experiences that define the process of software-assisted subjectivation in Facebook, as manifest in the specific way in which friendship is established. The various friendship experiences discussed below are not

meant as an exclusive list of human-machine becomings in and through Facebook. Rather the different steps and stages, including the registration process, the making of a profile, finding friends, as well as initiating, maintaining, and performing friendship should be read as various ways in which friendship becomes infused and augmented by software. In the process of experimenting with the various software features and functionalities related to Facebook friendship, my observations are based on my own personal Facebook profile registered on my real name, as well as a dummy profile that I set up in order to trace the Facebook-experience from the start. The dummy profile was created on April 1, 2011 in order to record the various steps and ongoing efforts made by Facebook to turn the user 'Carola Andrea' into a friend and active participant of the platform. I will begin my examination of 'algorithmic friendship' on Facebook by starting with what happens once a user registers and constructs his or her user profile.

One of the first things I did with the dummy profile 'Carola Andrea' was to add some friends. Thus I asked ten of my own friends to add 'Carola Andrea' as their Facebook friend. Facebook immediately took what little information they had about Carola and her new friends to help her fill out her profile (see figure 5). As such Carola was encouraged to take her friend Georg Kjøl's example to expand her profile.



Figure 5 Facebook welcome. Screen shot April 1, 2011.

Upon registering on Facebook, the user is instantly faced with the imperative to add friends. Compared to what can be referred to as a *Aristotelian conception of friendship*, as something rather precious that one cannot have with many people at once (Aristotle, 2004: 168), Facebook promotes the total opposite. One of the first suggested steps in the sign up process is to connect one's e-mail account with the new Facebook account, in order to allow for the synchronisation of existing contacts. In many cases the software already knows the new user. Based on the logic of the database, there is a great chance that Facebook already has some data stored related to the new user. With over 800 million Facebook users one might in fact exist on the platform by virtue of others having provided data about you, whether knowingly or unknowingly. Although one might think that ones Facebook-existence starts with setting up a profile, most users are 'ghosts in the machine' just waiting to come alive. This ghostly existence may for instance take the form of being 'tagged' in photos uploaded on Facebook by others, or 'checked into' places with others. Once a user has confirmed his existence by signing in, he is prompted to start filling in the template of the personal profile.

The profile is illustrative of promoting what Lisa Nakamura has described as 'menu driven identities' (2002). Nakamura discusses the logic of online forms that give little room for constructing identity in other ways than those already defined by the system. A menu driven identity, according to Nakamura, is a form of stereotype, making it easier to categorise, classify and sort people. As much as these stereotypes may have worrisome social, cultural and political implications (Nakamura discusses this in the context of race), the template driven identity promoted by Facebook and other social networking sites has a much more goal oriented purpose. Users' identities need to be defined within a fixed set of standards in order to be compatible with the algorithmic logic driving these software systems. If users could freely choose for themselves who and what they wish to say about themselves, there would be no real comparable or compatible data for the algorithms to process.

As anyone who has tried signing up on Facebook without adding friends can attest to, what drives Facebook are the friendships forged between users. How friendships are forged depends primarily on two separate, but interrelated, aspects, *findability* and *compatibility*. That is, how accessible and findable are you, and how compatible are you as a friend to others and vice versa. Default and privacy settings are important

features in regulating the desired flow of connectivity facilitated by the database of 'friends'. Users may choose and customise their privacy settings on Facebook, indicating and regulating the amount, nature and access to specific types of personal information. The platform itself configures personal profiles for connection by setting the default in the basic privacy settings of users personal profiles to 'everyone'. The privacy settings explicitly state that changing the defaults will 'prevent you from connecting with your friends'; conversely by keeping the default you will 'help' your friends from all spheres and passages of life 'to find you'. These privacy and default settings demarcate Facebook as a friend-collecting tool, geared towards optimising the friend recommendation algorithms for social graph enhancement.

Friending

One important mechanism for finding friends is the 'People you may know' (PYMK) feature that operates on a 'friend of a friend approach' (Chen et al., 2009). Friend of a friend (FOAF) is a common algorithmic technique for modelling friend recommendations online. This algorithm is based on the idea that if Anna is a friend of Tina and Alice is a friend of Tina, Anna could be Alice's friend too. As researchers on the data mining team of MySpace tellingly declared: 'Similar to real life, finding good friends is not easy without the help of good recommendations' (Moricz et al., 2010:999). A good and potential friend, according to the FOAF algorithm, is one that already shares a friend with you. Thus, the probability for someone to be recommended as a friend to another user increases the more friends these two people have in common, conceptualised in the attribute 'mutual friends'. Whereas most philosophical accounts of friendship view shared activity as the basis for friendship (Helm 2010), Facebook can be argued to put shared friends at the centre of friendship formation. The deployment of 'mutual friends' as a compatibility measure is clearly an important rhetorical strategy used in Facebook to suggest and amplify friendship initiations. Mutual friend counts appear on many different levels throughout the system. For instance, one encounters mutual friend displays whenever one goes to a friends' user profile, whenever one hovers over a friends' hyperlinked name with the mouse and whenever one receives a 'friend request'. Within the PYMK feature these mutual friends arguably play a decisive role in signifying compatibility for friendships.



Figure 6 People you may know. Screen shot from November 18, 2011.

Along with the smiling faces or otherwise flattering visual self-presentations manifest in the profile pictures, the mutual friend count functions as an implicit argument for why it makes sense for a user to add the other as a friend. Here we may begin to see the subtle ways in which algorithms can be considered actors in the sense that they prompt action, do things, or in Latour's sense: 'make a difference' (2005:154). Finding friends and forging connections is made easier with the help of algorithms. How many friends you have in common is used as the primary measure for friend compatibility. Once a user has found a certain number of existing friends and added them to their network, being on the platform becomes more meaningful.

It is again, interesting to note how the number of friends constitutes the basis for 'leading a good life' on the Facebook platform, in stark contrast to the Aristotelian idea of human flourishing realised through 'virtue friendship', where genuine friendship is about quality not quantity and loving the friend for the friend's own sake (Aristotle, 2004). On Facebook, the most direct way apparently realizing a 'good life', or meaningful existence, goes through the accumulation and number of friends. This is the law of network effects upon which social networking sites hinge. The more people are using it, the more useful it gets.

While the probability for someone to be recommended as a friend to another user by the algorithm increases the more friends these two people have in common, the number of 'mutual friends' also increases the probability of users accepting friend requests – even by strangers. In order to test the PYMK feature with the 'Carola Andrea' profile, I conducted a small experiment in 'friending' (adding people to ones network). On November 9, 2011, Carola, the dummy profile I set up, sent out friend requests to 40 random and unknown Facebook users (20 female and 20 male). The names were chosen using common American and Canadian female and male first names and the most common surnames. Facebook was then searched for these common anglophone names, and a friend requests were sent to the user who figured on the top of the list. After three days, six out of 20 people had accepted my friend request. After about five days, 13 users, or 32 per cent, had accepted my friend request.

On November 14, 2011, I sent out new friend requests, this time to friends of Carola's new friends. I used the PYMK tool to select two friends of the first seven people who had initially accepted Carola's friend requests, as these were the ones that the PYMK algorithm was geared towards. Selecting one random female and one random male friend of each of these initial seven friends, altogether 14 new friends requests were sent out saying we had '1 mutual friend'. After a few days 10 out of 14 people, or 71 per cent had accepted my friend request. Finally I sent out 10 new friend requests on November 21, 2011, again using the PYMK tool. This time I selected one person who had both the first set of accepted friend and the second set of accepted friend in common, so that the friend request would say '2 mutual friends'. All of these 10 people I sent a friend request to accepted it after a few days. Though the number in my study was small, it is possible to claim that the mutual friend indicator indeed increases the chance of positive 'friending'. While 1/3 of people accepted without a mutual friend, more than 2/3 accepted with only one mutual friend, and in this case everyone accepted with two mutual friends. It is also interesting to note how, out of a total 33 people who accepted Carola's friend request, only 6 decided to 'unfriend' her.⁶⁵

⁶⁵ Only 1 user out of the initial 40 sent Carola a private message asking if they actually knew each other. I responded to this user telling her that I in fact was a researcher who wanted to see how users respond to friend requests and to test the PYMK tool. I used the opportunity to

Performing friendship

How does the platform orchestrate friendships once formed? What possibilities for developing and performing friendship does the software offer? There are two principle features designed for engaging in practices of friendship on Facebook - the personal profile and the News Feed. Algorithmically driven, the News Feed displays an edited view of what one's friends are up to in an order of calculated importance, with the most important updates on top of the feed. The mechanisms at work in displaying the most interesting news about a user's friends and their actions, as I have already discussed in the previous chapters, are the EdgeRank and the GraphRank algorithms. Whereas EdgeRank passes judgement on the importance of *every* interaction related to the Facebook platform, Graph Rank, as will be recalled is a subset of EdgeRank geared specifically towards aggregating meaningful patterns out of users' interactions with applications. Every action and interaction connected to Facebook, be it a status update, comment on someone's photo, or 'like button' clicked, may become a story on someone's News Feed. Not every action, however, is of equal importance, nor is every friend for that matter. As such, algorithms are key to sorting the amount of data produced on the platform at any given time and to organising it into a meaningful stream of information for the respective user.

Everything and everyone that shows up in the News Feed have already gone through a selection process guided by the EdgeRank algorithm, which essentially decides 'which of the things your friends say should show up in your News Feed' (Madrigal, 2010). Friendships on Facebook are continuously measured, examined, and augmented by the software. EdgeRank does not just decide which stories should show up, but also which friends. As such, 'the secret-sauce algorithm is able to mystifyingly keep that dude from high school from continuously popping up in your timeline' (Blue, 2010). The power of the algorithm becomes apparent in its capacity to make certain people more visible than others. Thus we might say that the underlying software always already intervenes in the practices of friendship by selecting which friends a user should pay attention to.

ask her about her personal policy towards accepting friend requests on Facebook. Her response: 'I always check if the person has mutual friends'.

The power of EdgeRank lies not just in its capacity to define certain regimes of visibility in terms of assigning greater weight to edges that generate a higher degree of user participation. The productive power of algorithms also becomes suggestive of how friendships in their ideal form should be performed. As such, Facebook conditions the range of possibilities and modes of becoming a friend as governed by algorithmic forms of visibility, configuring the ways in which friendships are allowed to unfold within the boundaries of the platform.

One of the basic tenets of friendship as we know it from outside social media, is that it requires maintenance. Individuals must continuously work on developing and nourishing friendship, requiring a *repeated decision* to keep faith and mutual reinforcement (Derrida, 2005: 15-16; Webb, 2003: 122). In a context where the average Facebook user has 130 friends (Facebook, 2011), maintenance becomes hard and time consuming. However, several software mechanisms are put in place to help users nourish their friendships. For example, the now-pervasive ‘like button’ constitutes an important actor for expressing and articulating friendship. Initially, the ‘like feature’ was introduced as a social endorsement feature. No longer having to comment on a friend’s status update about a nice restaurant or new job saying things like ‘awesome’ or ‘congrats’, Facebook made paying attention to friends a one-click sentiment. As a token of ‘phatic communication’ the ‘like feature’ signifies the most cost-effective way of maintaining and performing friendships on Facebook.

Similar to how the ‘like’ feature can be said to activate important relational impulses, many other features on the Facebook platform are geared towards activating presumably important aspects of friendship performance. For instance, the friend birthday reminder feature counts as one of most important activators, as it spurs interaction on an ongoing basis. Having shown how various software actors interfere with and augment the various dimensions of friendship as it articulates on the Facebook platform, I now turn to the ways in which algorithmic friendship relates to more traditional accounts of friendship.

Towards an understanding of algorithmic friendship

Software, I argue, configures friendship online in a number of new and interesting ways. By encoding values and decisions about what is important, useful and relevant,

and what is not, software restricts certain activities by making others possible or impossible (Lessig, 1999). As I have shown, this becomes apparent when considering the multifaceted ways in which software elements, including the database, interface, privacy settings, algorithms, and code, to use Latour's formulation of nonhuman agency: 'authorize, allow, afford, encourage, permit, suggest, influence, block, render possible and forbid' possibilities for action (2005: 83). How then, can we understand the conceptualisation and construction of friendship via the Facebook assemblage?

One of the basic assumptions underlying the constitution of friendship in Facebook is the idea of *sameness* or *similarity* as a foundation of friendship. For example, the 'People you may know' algorithm introduces (or reintroduces) people to each other, based on a rather safe conception of similarity. The friend of friend approach thus 'represents a subtle form of limiting access to *difference*' (Elmer, 2004: 40). This logic hints at the politics of software in that an algorithm always selects and reinforces one ordering at the expense of others (Mackenzie, 2006:44). On the one hand, the PYMK algorithm conforms to traditional conceptions of friends as people who are somehow like us. On the other hand it also critically challenges a widely cited definition of friendship as 'voluntary relationships, largely free of structural constraints and based on equality' (Allan, 1989:1). Users are constantly encouraged and prompted to take certain actions, including befriending people. The 'people you may know' feature contained on users left-hand column functions as a constant reminder that there might be even more friends out there waiting to be added as friends. While the traditional notion of friendship highlights the voluntary and durational aspect of becoming friends and becoming friends anew, the software, one may claim, encourages and functions as a suggestive force that 'pushes' users to connect with the people he or she may already know according to the algorithm.

The subtle ways of software can thus be seen in the ways in which algorithms and databases pushes, reminds and (re)introduces users to each other. This is not to say that friendships on Facebook are not of a voluntary nature. Rather, the software assists in making friendships happen in the first place. Let's for a brief moment return to the notion of the ghostly Facebook existence mentioned earlier, and the ways in which the software may already know the user, even before signing up for the first time. This kind of 'technical remembering', what Stiegler calls 'tertiary retention' (1998), can be seen in the way that synchronising one's Facebook account with an existing e-mail

account allows the user to remember long lost social connections. When signing up on Facebook, users are prompted to import existing contacts by providing information about their e-mail accounts. Facebook will then provide information about who of the user's social connections are already on Facebook, subsequently suggesting that he or she adds them to their social graph. The accumulation of subjects into the Facebook database also feeds into the friend finding algorithm, constituting the grounds on which people are assessed for their compatibility as friends. Software thus enables remembering.

People we do not think about, people we might not remember or people we might not necessarily consider friends, continuously show up in the right hand column of our personalised News Feeds. A professor from another department, an acquaintance from ones student days, or simply a distant cousin on ones mother's side of the family are all candidates for possible new friendship connections dug up from the database by the PYMK algorithm and other memory aids. In an otherwise timeless system, Facebook seeks to simulate memory to promote a sense of community, commitment and responsibility. Various software features are put in place in order to activate friendship impulses.

For example, in late October 2010, Facebook introduced a feature called 'See friendship' that offers a way to 'view' the connection a user has with another Facebook user on an aggregated 'Friendship page' (see figure 7). This page can be found by following a link from underneath 'relevant Wall posts' (Kao, 2010). The 'See friendship' tool gathers data shared by two Facebook friends, including photos in which both are tagged, wall posts and the comments exchanged between them, events that both have attended in the past, how many friends they have in common, liked topics to 'tell the story of friendships' (Kao, 2010). This aggregated data generated by the digital traces left by users' online activity is subsequently visualised in the same manner as a regular Facebook user profile. This peculiar software-aggregated view of two friends' shared history on Facebook ultimately aims to 'bring back memories, conversations and times spent together' (Kao 2010).

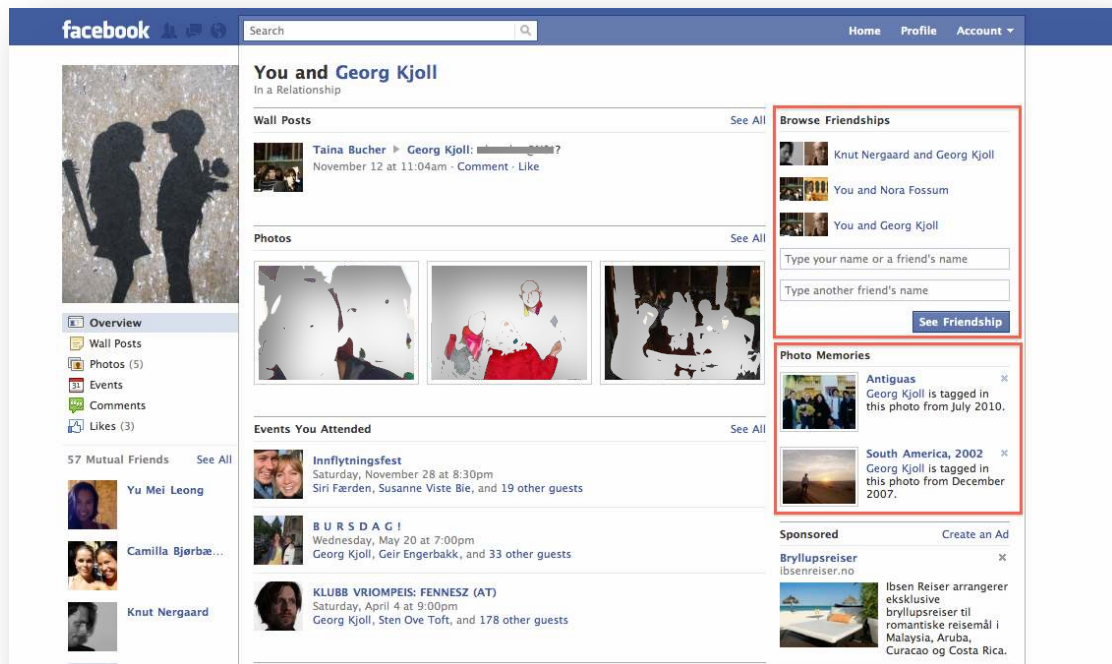


Figure 7 'See friendship' page. Screen shot from November 7, 2010.

The Friendship page is a form of information visualisation that seeks to generate interaction and more activity. The page provides both a view of the apparently long lost digital traces of a friendship, and allows for more browsing and activity through software features such as the 'Browse Friendships' and 'Photo Memories', that are placed in the left-hand column of the page. Through features like this, Facebook seeks to induce and simulate the emotional and intimate connections seen as a defining feature of friendships. As Facebook friends are not a set of distant strangers but rather people we already know, there is a certain responsibility on behalf of users to nurture and maintain these relations. The system supports this by prompting users to say hello, write on a new friends' wall, etc. What becomes apparent is how the software does not leave the users alone. Rather, Facebook needs to be seen as an active agent participating in the performance of friendship.

Traditionally, friendship has been thought of as an exclusive social relation. As Aristotle suggested, 'it is impossible to have friendship, in the full sense of the word, for many people at the same time, just as it is impossible to be in love with many people at the same time' (2004: 168). Above all, friendships are seen as requiring the time of continuous nurture and care (Aristotle, 2004; Derrida, 2005), meaning that

friendships are not static relations, but constantly evolve over time. Requiring ongoing affective engagement, folded in different temporalities and rhythms of repetition, memories and anticipation, the nature of friendship is fundamentally ontogenetic, or always in becoming.

Arguably, the ontogenetic nature of friendship becomes nowhere as apparent as it does within the context of Facebook. Users willingly and repeatedly *like*, *comment on*, and *tag* each other, creating the flow of attention needed to give the impression of continuous commitment. As EdgeRank is geared towards interaction, users who do not participate get downplayed, while users who frequently comment, like, and share are made more visible. In curious ways, Facebook reinforces the exclusivity of friendship and the classic conception of quality over quantity. However, contrary to the Aristotelian virtue ethics, quality in Facebook is measured on the basis of quantity. Therefore, it is the amount of engagement on the platform that becomes decisive for whether or not a user is regarded as a ‘good friend’ by the algorithm, and thereby made visible. The paradox here is that while EdgeRank reinforces the notion that we may only engage in a handful of friendships at any given time, it does so by encouraging users to accumulate friends. On another level, though, exclusivity is done away with completely.

Exclusivity and the notion that we must choose how to divide our time between friends, is inscribed into the EdgeRank algorithm itself. As already suggested, software aids us in deciding with whom to spend our time and on whom to focus our attention. Herein lies the power of the EdgeRank algorithm to determine users’ presence on News Feed. Friends embody different levels of interestingness, governed by algorithms. EdgeRank ‘distributes the sensible’ by revealing who can have a share in the community of friends, based on what they do (measured in the amount and nature of participation), the time spent on the various friendship practices (measured in frequency and recency), and the space in which the activity took place (some spaces are given more weight, such as talking to each other via Facebook chat or messages).

We can see another paradox with regards to exclusivity if we consider Facebook friendship in light of another traditional conception of friendship, as one of a private and intimate relation between two persons (Hays, 1988: 395). Far from being a private and intimate relation, the Facebook architecture is set up in such a way as to

make friendships public, as something that engages the whole network. Friendships are not exclusive to two people, as they have become a matter of the network. Friendships on Facebook are on display and made to engage friends, as well as friends of friends. However, Facebook needs to balance the private and public aspects of friendship. While designed to engage the network, Facebook also hinges on users' having a sense of intimacy, so as not to flatten out the meaning of the friendship relation. Maintaining the notion of friendship as something to be cared for and nurtured is therefore extremely important for a commercial social networking platform like Facebook, where friends are a valuable currency.

Precisely because friendships signify something exclusive, a social relation that implies trustworthiness, friends can be used for commercial purposes. The semantic-technical expansion of Facebook to other parts of the Web, through the implementation of the Open Graph protocol and the 'Like button', is based on the perceived commercial value of friends. Liking content across the Web provides valuable data about people's affinities. The data captured by a 'Like button' is above valuable in the sense of providing a means of accessing people's friendship networks. As such, the Open Graph and the 'Like button' directly intervene in the production and circulation of meaning, by framing friendship as a currency that can be used for commercial purposes.

When liking content across the Web, a link to Facebook is established, making the action into a potential story on the News Feed. Again, not every friends' affinities and likings are of equal important to every user. Through the close interconnection and interoperability between protocols, code and algorithms on Facebook, the software 'decides' whose 'likes' might be worth more. Exactly how valuable a 'like' can become depends on the EdgeRank.

The 'Like button' also thrives on the assumption of sameness or similarity, meaning that users desire or like the same things that their friends do. Tellingly, one of the largest consumer trend firms in the world highlighted what they called the 'the f-factor' in their May 2011 trend report (Trendwatching, 2011). The f-factor refers to the power of friends in consumers' purchasing decisions today. The report bases its claim precisely on the assumption that we trust our friends and think like our friends, turning friends into the most relevant recommenders. For example, the assumption of sameness embedded in the 'like button' is amplified by the jeans company Levi's,

when they tout what they call ‘like-minded shopping’, or the travel site Tripadvisor.com, which refers to ‘trusted hotel reviews’ produced by Facebook friends.

Moreover the commercialisation of friendship can be seen in the move towards using friends as *ads*. Through a feature called ‘sponsored stories’, Facebook generates ads out of friends’ online actions. Displayed on users’ right-hand News Feed column, ‘sponsored stories’ blends in with the rest, turning ‘friends’ actions into promoted content’ (Parr, 2011). When John, for instance, likes Starbucks on a website that is semantically connected to Facebook, a story can be published on his friends News Feeds, saying that he endorses or recommends the brand. As such, businesses and organisations can leverage on the principle of word of mouth, arguably representing the most powerful way of recommending things.

 ...'s birthday is today
 **Create Event**

Find More Friends

Carola, More Friends Are Waiting



These 3 friends found their friends using the friend finder. Have you found all of your friends? Give it a try.

Your Email

Email Password

Find Friends

 Facebook won't store your password.

People You May Know [See All](#)



3 mutual friends
 **Add Friend**



2 mutual friends
 **Add Friend**

Sponsored Story [See All](#)



... likes Nissan Versa.



Nissan Versa
 **Like**

Figure 8 Sponsored stories. Screen shot from November 19, 2011.

Friendships on Facebook, then, are shaped through the particular ordering of the social – the distribution of the sensible – in and through software, and especially the algorithms orchestrating News Feed. Friends embody different levels of interestingness as a consequence of the social sorting mechanisms taking place at the level of the algorithm. Users are given the impression of constant activity, as those friends whose updates are most frequently commented upon are given priority over those whose updates are only rarely commented upon. Thus, the algorithmic logic based on popularity prioritises ‘important’ people. Facebook rewards the ‘influencers’ with visibility on the basis that they attract a lot of interaction.

The probability of attracting the attention of others will in turn increase the more connections one has. The more friends, the better the chances of receiving comments or likes on ones actions on Facebook. The more interaction a user is able to generate, the greater the probability of becoming visible on friends’ News Feeds will be. Collecting and accumulating friends is therefore not just a peculiar feature of online social networking, but also a necessity for getting on the News Feed. Not only does accumulating friends enhance a user’s chance of becoming visible on the News Feed; it also enhances the chance of becoming a potential friend to someone else. That is, a ‘user with a wider social circle has a higher probability of friendship overlap and therefore may be recommended frequently to many different users’ (Daly et al. 2010, 302). The politics of algorithmic friendship becomes evident when one sees that the probability of being a recommended friend increases with a user’s popularity, and that it diminishes as a user has fewer friends. What the friend-of-friend algorithm essentially suggests, is that a desired friend is the person who frequents diverse social networks and has as many friends as possible.

What friends are for

Social networking sites are inherently about friendship. However, as this chapter has shown, Facebook does not merely facilitate friendship by providing a platform for existing friendships; it is also a ‘friendship maker’ (Wellman, 2004). Online friendship thus needs to be understood as a socio-technical hybrid, a gathering of heterogeneous elements that include both humans and nonhumans.

In an increasingly software-mediated world, a sensibility towards the active role of

non-human actors creating forms of sociality becomes imperative. Thinking of friendship as an assemblage – a relational process of composition – has offered me a way to critically scrutinise how software participates in creating, initiating, maintaining, shaping, and ordering the nature of connections between users and their networks. What, then, does the software suggest about the articulation of friendship on Facebook? What are friends really for? To conclude the chapter, I would like to draw attention to three points of divergence between more broadly held conceptions and cultural ideas of friendship, and the ways in which friendship is produced by the software.

First, friendships on Facebook are never only between two individuals. Rather, we need to think of online friendship as a relationship involving a plethora of actors, both human and non-human. Users do not simply send out ‘friend requests’ or accept them; software actors like the PYMK algorithm increasingly assist users in making these kinds of decisions. While friendships are still about selecting and making decisions about whom to devote one’s time and attention to, these decision-making processes have increasingly been delegated to algorithms. It turns out, then, that Derrida’s question about what a future would look like if decisions could be programmed (2005, 29), has found at least one answer in the sociotechnical construction of friendship on Facebook.

The programmability of friendship importantly helps orchestrate the various temporalities of the notion. For Derrida, friendship entails two primary modes of temporality - constancy and anticipation - understood respectively as ‘the repeated renewal of a decision to ‘stick with it’’ and as ‘the future that will be opened, which could not have occurred without the friend’ (Webb, 2003:122 & 136). Arguably, the quality of constancy is augmented, supported and induced by the software system. The software helps users ‘remember’ everything from friends’ birthdays, friends’ most ‘memorable status updates’, and the shared history of a friendship as manifest in the ‘see friendship’ function.

The algorithmic configuration of friendship hinges on the anticipatory logic of friendship. In the Friendship assemblage of Facebook, friends are used as key variables to calculate the probabilities of future actions on the platform. Friends’ preferences and actions thus constitute important information for the algorithmic, data-driven logic of Facebook. As the Facebook engineer Narasimhan puts it in a

recent tech talk, ‘recommendations are essentially a function of who you know and what they like’ (Facebook Engineering, 2011).

Secondly, friendships on Facebook break with some of our most engraved cultural assumptions about friendship, in terms of depending on shared and reciprocal activities (see Lynch, 2005: 189). Friendships in many philosophical accounts, are thought of as relationships that hinge on shared activity, reciprocity, mutual contact and joint pursuits (see Aristotle, 2004; Telfer, 1991). While the publicity of friendship on Facebook in some respect qualifies as a form of shared life, the algorithms suggest otherwise. True, the initiation of friendships on Facebook, with the help of the PYMK algorithm and ‘friend request’ feature, hinges on the notion of reciprocity.

However, when it comes to performing friendship on Facebook, the cultural logic of reciprocity and mutual contact are undermined by the operational logic of EdgeRank. For example, the fact that my friend Anna shows up in my news feed does not imply that I also show up on hers. As Josh Constine reports, ‘your average Facebook post only reaches 12 per cent of your friend’ – ‘you’re not unpopular, it’s just the nature of the news feed’ (2012). Because of this, one needs to be wary about treating the News Feed as a kind of public sphere, or common space of living together in the sense of Aristotle. In a algorithmically regulated space like News Feed, access to the community and the right to have a say and be heard, is increasingly defined by a person's popularity, sociality and social status. Friendship on Facebook turns into the kind of illusion that Derrida (2005) speaks about, in that it many ways functions unidirectionally, as opposed to reciprocally. In other words, while news feed may produce the ‘illusion’ that I have an ‘active’ relationship with my friend Julie by making her repeatedly visible on my news feed, Julie might for all I know feel the opposite. While for Aristotle, reciprocity is the glue of all friendship (Vallor, 2011), friendships on Facebook needs no glue; they function themselves as glue. As Facebook expands by the help of the Open Graph protocol into other parts of the Web, friendships are the glue between Facebook and external websites. In a networked society, friendships become attached to a user’s digital persona that they cannot escape.

Thirdly and finally, we can claim that, contrary to the notion that friendship is something in which we freely choose to engage (Allan, 1989), friendships on Facebook loose some of its voluntary character. As the presence of the other cannot

be escaped in an environment of nodes and edges, the connections we forge with other people may have real consequences, as the conditions of the intelligible and sensible is increasingly calculated on the basis of who our friends are, what they have done and how many of them there are. In this sense, the Facebook friend resembles the philosophical idea of the friend as a 'second self' (Aristotle, 2004: 189) to a certain extent. However, whereas the Aristotelian conception of 'second self' hinges on respect, love and wishing a friend well for the friend's own sake (Aristotle, 2004: 162), the Facebook friend as a 'second self' needs to be understood in a much more literal sense.

Whatever action a user's Facebook friend performs on the platform, say 'checking-in' at a restaurant or 'liking' a brand, it will automatically get associated with the user. A user's digital identity is thus collapsed with that of his or her friends. The implications of this can be seen as far-reaching, given that friends become a primary means through which the production and occlusion of information can be 'programmed'. Contrary to a conception of friendship as something created between equals and free of structural constraints, friendship in the age of 'programmed sociality' needs to be understood as a process of sociotechnical negotiations between users and software.

Recognising the importance of software as a participant in friendship relations raises a question about the potential control of these relations. What and whom friends are for, thereby becomes one of the most pressing questions in media research today, as friends on Facebook increasingly constitute the economic cement of our current information ecosystem.

In this chapter I have shown how software intervenes in the formation of friendship and how software therefore needs to be considered as an important actor alongside humans in the construction of sociality online. The Facebook platform not only activates certain relational impulses framed in terms of friendship, the platform also hinges on the fact that these relationships are maintained and cared for in manners that purport ongoing participation. In the next chapter I continue my investigations into the ways in which software can be said to act as a catalyst for participation. Whilst I have focused entirely on the Facebook platform up until now, in the next chapter I turn to the case of Twitter and its application programming interfaces (APIs). I see the Twitter APIs as a particularly fruitful site at which software can be analysed for its capacity to 'hang or concatenate relations together' (Mackenzie, 2006:11). Thus, the

aim of the next chapter is to investigate the circulatory and contested existence of code through the case of the Twitter API by paying attention to the multiple participants involved, their interests and desires. What kinds of relations are assumed to be desirable and whose interests are emphasised, and devalued? ⁶⁶

⁶⁶ An earlier version of Chapter 7 is forthcoming in *Television & New Media*. The article 'The Friendship Assemblage: Investigating programmed sociality on Facebook' has been published online before print, August 24, 2012.

Chapter 8. The Twitter APIs: Objects of intense feeling

One of my claims thus far is that software is a slippery and convoluted thing. In the previous three chapters I have been concerned with Facebook and the ways in which software participates in the shaping of particular attentive and participatory subjectivities and ways of being together, through what the platform calls friendships. Through what I have called a technographic approach, different modalities of programmed sociality have been examined, mainly by examining forms of protocological and algorithmic power. In this last chapter, I turn to the case of Twitter and its thriving third-party developer ecosystem organised around the platform's application programming interfaces (APIs). Seen as protocological software objects, the question here is how the APIs form and hold relations together and how can we understand the coordinative work that protocols do?

When faced with things that are variable and slippery in nature such as software, one way to proceed is to consider the multiple concerns that software encompasses. In this chapter, I therefore explore a more specific example of such a convoluted gathering, the Twitter application programming interfaces (APIs). Not only are APIs interesting objects to study in and of themselves; largely ignored by media and communication studies, the Twitter APIs offer a way in which to study the means by which software gathers different actors, and how 'code hangs or concatenates relations together' (Mackenzie, 2006: 11).

The Twitter APIs and its third-party developer ecosystem have arguably played a decisive role in the popularity and success of Twitter as a microblogging system. APIs are 'specifications and protocols that determine relations between software and software' (Cramer and Fuller, 2008: 149). Many social networking sites and Web 2.0 platforms provide APIs so that developers and other interested users may access and use some of the data and functionality that the software has to offer. APIs have made it possible for developers to build new software products on top of already-existing software. As a consequence, we have seen the rise of so-called mashups, web applications that combine data and functionality from two or more sources/software programs, not to mention the explosion of apps, software applications designed to run on smartphones and tablet computers.

In this chapter I will investigate the Twitter APIs as entities that gathers multiple actors, interests, and ideas – not always in ways that stabilise the entity, but rather as conflicting relations that make software a contested and very visible object. I will argue that Michel Serres’ concept of the quasi-object provides a helpful analytical framework in which to understand how APIs catalyse new modes of collective associations. Serres uses the term *quasi-object* as a way to describe the ways in which sociality is played out in conjunction with objects, how objects act as catalysts for various social relations and actions (Serres, 1982; 1995). Seeing APIs as reminiscent of quasi-objects helps to see how software plays an influential role in organising and regulating the playing field of sociality in the context of Twitter.

Specifically, I focus on the third-party developer ecosystem as a means to explore what I see as an important site at which collective associations of humans and nonhumans coalesce around matters of code and software. Empirically, I draw on material I have collected that is related to the Twitter APIs, including documents, online observation, and online interviews with some of these developers (see Chapter 4). Although I have chosen to quote selectively from the interview material, all of it has been crucial for gaining an understanding of the ‘blips’ of software that the developers themselves seem to care about; that is, the contested site of the Twitter APIs. ‘Blips’, as Matthew Fuller sees it, are certain events in software, operations, and regimes ‘at which interrelations, collaborations and conflicts can be picked out and analyzed for their valences of power, for their manifold capacities of control and production, disturbances and invention’ (2003: 32).

Seeing APIs as blips or events in software implies an understanding of APIs as agential. APIs are thus understood as actants that have the capacity to shape how people live their lives. As Kitchin and Dodge suggest, ‘software divulges and affords agency, opens up domains to new possibilities and determinations’ (2011: 39). How, then, can we begin to understand the ways in which the Twitter APIs make a difference to the everyday lives of programmers, how they produce different relations and experiences, and how they participate in shaping the material-discursive spaces of social networking sites? One way of approaching these complex questions is to acknowledge the manifold relations that permeate APIs, or as Mackenzie suggests, ‘the forms of contestation, feeling, identification, intensity, contextualizations and

decontextualizations, significations, power relations, imaginings and embodiments' (2006: 5).

In my e-mail interviews with third-party developers who work with the Twitter platform, it became evident how affective and culturally 'invested' software really is. Borrowing Mackenzie's words, I argue that the Twitter APIs constitute an 'object of intense feeling' (see Mackenzie, 2006: 71). The Twitter APIs do not merely draw programmers together into communities of practice, alliances, and collaborations, ultimately enlisting different actors to engage in different types of work. The APIs also regulate the playing field in regard to what can happen where and when, what can be built technically, and policy-wise by constituting an infrastructure for innovation. APIs constitute the condition of possibility for many new coding practices and cultures on the Web, a situation that is perhaps most clearly illustrated by the many apps that make use of social networking data. APIs are also productive of new subjectivities and forms of governance around software development in an age of 'cultural work' (Gill and Pratt, 2008). APIs are mobilised by social media companies as governmental techniques oriented towards creating and controlling the boundaries of cultural production, as manifest in third-party software development and its subjective processes.

While it is not my primary concern here to dwell on questions of labour or creative work, the broad range of scholarly work on what has been described as the 'new conditions of the creative industries' certainly provides an important backdrop for understanding the processes of innovation and software development in the context of social networking sites (see for instance Ashton, 2011; Banks and Deuze, 2009; Gill and Pratt, 2008; Neff et al., 2005). According to these commentators, cultural work in neoliberal societies is increasingly structured around the discourse of enterprise (Ashton, 2011: 315). For du Gay, enterprise culture became the predominant business paradigm during the 1990s. Enterprise culture promotes values such as self-reliance, personal responsibility (for instance for acquiring the right skills, or for one's own failures and successes, etc.), and risk-taking (Marwick, 2010: 298; du Gay, 1996). Rather than investigating the extent to which such discourses are reproduced or whether web developers live up to such claims about enterprise culture, I concur with Gill and Pratt in their claim that we need to pay more attention to 'the meanings that cultural workers themselves give to their life and work' (2008: 19). As such, this

chapter locates an understanding of the productive power of software in an analysis of the Twitter APIs and its developer community, and investigates the types of connections APIs are able to forge, and the possibilities for actions that are offered.

Following Mateas and Wardrip-Fruin, the question is how software means something in different contexts and how to account for specific processes of meaning making (2009: 4). Here, I want to approach this question by following some of the strategies proposed by Marcus in his description of a multi-sited ethnography (see Chapter 4; Marcus, 1995). The aim is not so much to make general claims about the ways in which software is meaningful, but rather, as Mackenzie observes, to help us understand the encounter between discourses and processes of software production, and a group of actual programmers (2006: 143). In this sense, Marcus' account of a multi-sited ethnography, as introduced in Chapter 4, offers some useful tools for tracing the various 'chains, paths, threads, conjunctions and juxtapositions' of the Twitter APIs. These strategies involve 1) 'following the people', as in the Twitter third-party developers, 2) 'following the thing', as in the APIs themselves, their ontology and their circulation into different sites and locales such as online discussion threads and technical documentations, and 3) 'following the conflict' – meaning specific sites or occasions where the objects of study become particularly visible, debated, and contested. With the help of Latour, the point of departure for such an associative inquiry is to examine the multifaceted ways in which the Twitter APIs 'might authorize, allow, afford, encourage, permit, suggest, influence, block, render possible, forbid, and so on' (2005: 72).

This implies not merely viewing APIs as 'specifications and protocols that determine relations between software and software' (Cramer and Fuller, 2008: 149), but also in the sense of the quasi-object, as protocols that shape the relations between multiple actors, including humans and nonhumans. If, as Mackenzie suggests, 'much hinges on how code hangs or concatenates relations together' (2006: 11), the question becomes how the Twitter APIs hold relations together and which principles of connections are perpetuated. What are the APIs capable of, and how do they activate certain relational impulses?

Application Programming Interfaces

In order to understand the concept of application programming interfaces, it might be useful to point out how APIs are instantiations of basic software engineering principles (de Souza and Redmiles, 2010). APIs are the interfaces that make it possible for different software modules to interact. The principle of ‘modularity’ in software design refers to the ways in which software is usually made up of various components or modules. This reduces the complexity of systems, and makes the software more manageable (see Baldwin and Clarke, 2000).⁶⁷ Lego bricks may serve as a helpful analogy here. Lego bricks are modules that can be put together in an indefinite number of ways and reused at will. Not all Lego bricks fit together, but when they do, they work together – or interoperate – to build everything from simple to complex structures. In much the same way as Lego allows for different parts of a construction to be built separately, the division of code into modular units makes it possible for programmers to develop parts of a software system independently, and to work in parallel. Most importantly, modularity makes it possible to separate concerns, through what Parnas (1972) called ‘the principle of information hiding’. As de Souza and Redmiles exemplify, the principle of information hiding holds that ‘software modules should hide implementation details that are likely to change and expose only aspects that are less likely to change’ (2010: 446). APIs, then, are a common instantiation of the principles of modularity and information hiding that enables two software components or applications to ‘talk’ to each other.

⁶⁷According to Baldwin and Clarke (2000), the first modular design of a computer system was IBM’s System/360 introduced in 1964. The implications were huge. For the first time, rather than software being locked down to a particular hardware platform, it could now be used across the different machines. Before the System/360 was introduced, programming and machine operability were tied to the specific design of a particular machine, which meant that knowledge of how to operate and program one machine was not necessarily transferable to other machines. Because the designs were so specific, it also meant that ‘each new system had to be designed from scratch’ (Baldwin and Clark, 2000: 171). For instance, as Campbell-Kelly points out, ‘in 1960 IBM had no fewer than seven software-incompatible computer architectures. Unique operating systems and utilities had to be developed for these platforms, and each language processor had to be recoded’ (2004: 95-96). IBM’s efforts with System/360 were based on the recommendations put forward by the SPREAD Task Group, established by the company to tackle the growing discontent among IBM’s customers with the locked-in design approach and the incompatibility with other systems. In effect, interoperability and modularity became important design principles on all levels of the computer system.

In this chapter, I am concerned with a specific type of API, the web-based ‘open APIs’ that have become an important element of many Web 2.0 platforms today. Whereas the general concept of an API involves methods for interacting with a piece of software, an open web-based API refers to the kinds of methods that allow for programmatically accessing data and functionality via HTTP. The term ‘open’ might be a bit misleading, as these services do not open up their databases completely, but rather offer data in a highly controlled and regulated way. ‘Open’ alludes to the ways in which these APIs are made publicly available. Importantly, the general concepts of an API and ‘open APIs’ differ with regards to their genealogies. Whereas the former grew out of software engineering, the latter came out of a specific business context. As Bodle points out, open APIs became an important business strategy for Web companies ‘soon after the dotcom bubble peaked in March 2000’ (2010: 325). ‘Far from a risky business strategy, opening APIs was considered a sustainable business move to encourage the growth of a supportive ecosystem of third party developers, which could increase the value of a platform or web service’ (ibid.).

The API directory site Programmable Web currently (March 2012) lists 691 different mashups built using Twitter APIs. For example, there is TweepMap, an application that leverages both the Twitter and Google APIs to analyse the geographical location of a Twitter user’s followers, and plots them on a Google map. Or, paper.li, which mines a user's tweets and the tweets of that user's friends to find the top stories, and displays them as a daily newspaper. In my interviews with Twitter developers, three broad methods and reasons for using the APIs could be discerned: the specialised method of using Twitter data and functionality to build some kind of niche application; the general method that allowed access to as much of the data contained by Twitter as possible, often to build clients, or for data-mining and visualisation purposes; and what could simply be called a personal method to leverage the data. In the case of my informants, the specialised and generalised methods often went hand in hand with some kind of commercial motivation or desire to eventually make a successful business out of their software. For example, Jeremy uses the Twitter API to build a site that analyses tweets related to the travel industry only, while Jacob has developed and maintains a general Twitter client for Mac OS X. Oliver, moreover, describes the opportunities he saw in the API as follows: ‘I thought there was an unfilled niche, namely [a] lack [of] twitter clients with multimedia capabilities (i.e.

attaching photos, videos) on low-end phones'. The personal method, on the other hand, mainly speaks to using data out of self-interest. In this sense, Eric, another of my informants, got into the Twitter API because he wanted to republish tweets about Formula 1 – his favourite sport – on his own website, so as to 'provide a kind of live action and a basis for a discussion' with other fans.

The first company to offer a peek into its database was eBay, in 2000, followed by Amazon with the launch of Amazon Web Services in 2002. The idea was quite simple and reminiscent of long-standing efforts within free and open-source software development; namely, to outsource research and software development efforts to a potentially indefinite number of software developers. As Roush suggests, outsourcing software development in Amazon's case was essentially about letting others find 'cleverer ways of using Amazon's data than Amazon itself' (2005: 28). Thus, as CNN reported, 'by opening up parts of their platforms early and voluntarily, Amazon and eBay, and others increase their chances that developers and businesses will organize around their systems' (Schonfeld, 2005). This 'counterintuitive business strategy', as Roush (2005) described it, started to take off in 2005 when Google launched its Maps API. The outsourcing rationale can be seen clearly in Google's own announcement at the launch: 'If you like Google Maps, but think you could do something better, now's your chance' (Taylor, 2005). Today, most Web 2.0 companies offer APIs so that third-party developers can build new applications on top of their platform. For instance, the photo-sharing platform Flickr has one API, Youtube has two, and Facebook has as many as eight different APIs. Not only do APIs offer a way for programmers to more easily access some of the data that these companies have, but APIs have also become a useful way for these companies to extend their reach and growth across the Web. Perhaps most importantly however, as Bodle points out, APIs have established conditions for online sharing and participation, setting the boundaries for the competition over the control of social media flows (2010: 326).

Open APIs are interesting objects of study for a number of reasons. First, APIs constitute one of the most important software elements of Web 2.0. The whole notion of participatory culture is arguably built around the idea of users sharing and generating content online. APIs are responsible for the ways in which content becomes shareable across different websites and applications. Having played a decisive role in the success and growth of social media, APIs continue to be important

today as they lie at the very heart of the growing ‘app space’ powered by smart phones. Second, the opening up of software through APIs has had a profound impact on the nature of web programming and software development. By offering a way for outsiders to interact with the software service and data that is provided, APIs have opened up important new job and coding opportunities for programmers. Thus, APIs can be seen as affordances of sorts, providing possibilities for action by being suggestive of how they can and cannot be used, and ultimately governing participation in particular ways.

Despite the important role they play in the context of social media, very little academic research exists on APIs from a media and communications or software studies perspective. One notable exception here is Robert Bodle’s (2010) article ‘The Regimes of Sharing’, to which I have already referred. To a large extent, the existing academic literature on APIs comes from the field of computer-supported cooperative work and HCI, and is for the most part focused on practice and design. As de Souza and Redmiles point out, this research is aimed at ‘how to design APIs (Michi, 2009), how to evaluate their usability automatically (de Souza and Bentolila, 2009) or manually (Ellis et al., 2007), how to use APIs based on examples (Xie and Pei, 2006)’ (2009: 447). In addition to the academic context, APIs have been widely discussed in the popular business and technology press (see for instance Kirkpatrick, 2008; Metz, 2012; Parr, 2009).

Following Galloway’s (2004) notion of protocol as a management style, APIs can be seen as the management style of social networking data, governing access to and flow of these data, and organising the practices of coders and coding around these data. As we will see, APIs can be viewed as being comprised of a set of techniques for managing contingent relations that affects not only the technical level to which they pertain, but importantly influences how sociality is programmed or managed. This chapter is thus an attempt to help close the gap in the academic literature on APIs, and specifically in regard to the construction of collectives and sociality around these pervasive software objects. Given the powerful role APIs play in enabling the flow of data and information on the Web today, it becomes imperative for media scholars to engage with the many convoluted ways in which APIs bring together relationships of capital, control, and freedom.

The Twitter APIs

In this section, I turn to the specific case of the Twitter API. I will provide general background, and elaborate on how Twitter connects to the business logic described above. I then describe some of the technical specifications of the APIs, before considering the community of practice surrounding them in more detail. Drawing on data from interviews with Twitter third-party developers, the aim is to explore some of the ways in which APIs create meaning, and how they reveal and afford new possibilities and opportunities for action.

The microblogging service Twitter was launched in July 2006, and quickly became a favourite application in the social media industry (i.e. marketers, technology bloggers, industry people, and early adopters).⁶⁸ Twitter initially started out as quite a simple tool with only one basic feature, the stream, and one function, the possibility to write a status update. The purpose was to prompt people to disclose their thoughts and activities in real time by providing an answer to the question: ‘what are you doing?’.

⁶⁸ This is evidenced by the amount of press coverage Twitter received early on, as part of the mainstream media, but most importantly from authoritative technology blogs such as ReadWriteWeb, TechCrunch, and Mashable. Twitter also won the annual SXSW Web award in 2007, only six months after it launched.



Figure 9 Twitter as of November 27, 2006. Source: Internet Archive

In September 2006, only two months after its launch, Twitter published its Twitter API and made it publicly available. The message on the Twitter blog was quite clear: ‘we’ve exposed some of the inner workings of Twitter so engineers who want to creatively extend our functionality can do so’ (Twitter, 2006). This move, to open up their service to outside developers via an API, has been crucial to the success of Twitter as a company, software service, and social networking site. The idea of letting anybody with enough programming skills have a go at their API to access data and functionality became a key business strategy of Twitter, and a decisive factor for the growth and popularity of the service. As Biz Stone, one of the service’s three founders, declared in an interview September 2007:

The API has been arguably the most important, or maybe even inarguably, the most important thing we’ve done with Twitter. It has allowed us, first of all, to keep the service very simple and create a simple API so that developers can build on top of our infrastructure and come up with ideas that are way better than our ideas (Ammirati, 2007).

As we can see, the philosophy of letting others come up with good ideas and innovations popularised by companies like Amazon, eBay, and Google also became an important strategy for Twitter. Twitter’s sparse beginnings as a web service

quickly grew into an ecology of third-party applications and mashups. As ProgrammableWeb recapitulates:

Twitter's growth can be attributed to the Twitter API, which allowed the company to be on every mobile platform before it had an internal team building mobile apps [...] Even Twitter's search engine was built on Twitter's API. Originally called Summize, it was Twitter's first acquisition way back in 2008. The product, which was better than anything built internally at Twitter, is also available via the Twitter Search API (DuVander, 2012).

Almost from the very beginning, API usage generated much more traffic to the main site than did end-user engagement. In 2007, Twitter API traffic amounted to 10 times that of the Twitter site (Musser, 2007), while this number had reportedly increased to 20 times the main site by 2009 (Lennon, 2009). In 2010, ProgrammableWeb reported that a staggering 75 % of Twitter traffic came from third-party applications (DuVander 2010).

Little by little and steadily over time, Twitter has acquired the most popular third-party apps and clients that developers were making, and they were often made part of the Twitter core service. As Tony, one of my informants, elaborates when I ask him about the nature of software development in the age of open web-based APIs:

Software companies have to develop a very small part of the software and they get millions of developers FOR FREE (almost) all around the world, creating new improvements to the original software. The companies can even incorporate those modifications into the software if they become important enough, and the cycle keeps going round!

The story of how Twitter 'developed' a search tool as part of their core functionality, is a good example of Twitter's early reliance on the third-party ecosystem. Twitter had been working on a search tool called 'Track', which did not turn out to be the success they had hoped for. According to Twitter cofounder Ewan Williams, this was partly because they had released it too quickly, and had not figured out a sustainable systems architecture for it at the time (see Arrington, 2008). Instead of trying to fix the problem in-house, Twitter made a deal with Summize, the most popular third-party search tool on the market at that time. Twitter made the search tool part of its code, continued to offer the open API for search that Summize had originally provided, and employed all five Summize software engineers as part of the Twitter team (Arrington, 2008). Similar patterns of acquisition have occurred with other hugely popular third-party apps and clients, including the acquisition of Tweetie, a

Twitter app for the iPhone in April 2010, and Tweetdeck, the most popular desktop client next to Twitter itself, in May 2011.⁶⁹

Technical specs

Currently, Twitter has three different APIs: two REST APIs including one for search, and one for streaming. The REST API, which stands for ‘Representational State Transfer’,⁷⁰ provides a way for developers to access and manipulate Twitter’s core functionalities, such as the posting of tweets, viewing a user’s followers, and sending and retrieving messages. The search API lets the developer search for certain data by keywords, username, or location, as well as view historical trends. Finally, the streaming API pushes data to partners in near real-time. Basically, ‘open APIs provide information to third-party applications through ‘calls’, a technique of retrieving data on a server in the background, without disrupting the display and function of a web page’ (Bodle 2011: 322). The API documentation provides a list of methods that the third-party developers may use in order to build their own apps, websites, and clients on top of the functionality and data offered by the respective platforms. Twitter’s APIs are based off the HTTP standard. REST outlines a convention that ensures the API follows a consistent set of rules in accordance with HTTP (see Kincaid, 2010b; Parr, 2011a). The REST API gives endpoints that may be used to read timelines, post tweets, and implement numerous other functions. For example, the timeline resource provides nine different ways in which programmers can GET/receive a collection of tweets. Some of the other functions offered by the API include checking which users are members of which lists, receiving detailed information about the relationship

⁷⁰ Technically, web-based open APIs usually conform to one of two consistent architectural frameworks for implementation, REST or SOAP. Simple Object Access Protocol (SOAP), developed by Microsoft in 1998, is one of the most widely-used frameworks for building web services. While SOAP is a W3C-recommended web architecture standard, Representational State Transfer (REST) is not considered a standard so much as a style for designing networked applications by using simple HTTP. REST has to a large degree replaced SOAP as a standard for communication with other Web services over networks, which can mostly be explained by its simpler and more implementation-friendly nature. REST accepts many different message formats in contrast to SOAP, which is entirely bound up to XML as a standard message format. According to ProgrammableWeb, as of February 14th, 2012, 71 % of all APIs now use the REST protocol, compared to only 18 % using SOAP. See <http://www.programmableweb.com/apis>.

between two users, and receiving the 20 most recent favourite statuses by particular users.⁷¹

Importantly, APIs are highly-controlled gateways to data. Offering an API does not mean that all the data stored by a service such as Twitter is freely available to anyone who is technically literate enough to make a few HTTP requests. API calls are usually limited, to prevent full access to the data trove and to keep API management under control. In Twitter, the majority of GET requests are rate limited, while POST methods are not.⁷² Basically, the API controls the types of applications that can be built, and, by imposing rate limits, the extent to which data can be accessed and repurposed by third-party developers. While the limited access to Twitter data is free of cost to any interested party, access to the full ‘fire hose’ of data is only provided to a few select partners, such major search engines and the like. For data analysis purposes, more specific access can be bought from either of Twitter’s two data provider partners, Gnip and DataSift.

Community of practice

One of the striking aspects of Twitter is the community of practice that has formed around their APIs from the very beginning. At the time these interviews were conducted (August 2010 - August 2011), the Twitter third-party community was for the most part gathered in the ‘Twitter Development Talk’ Google group. The Google group was established March 2007 to provide a place where third-party developers and other API users could come together to discuss issues connected to the Twitter APIs. The discussions could revolve around purely technical difficulties or be more normative and deliberative in character, for instance discussions of Twitter’s terms of service and developer rules. As of August 2011 the group counted 12237 members, with an average of almost 700 messages posted per month during the first half of 2011. At most, the group received 2240 e-mail messages (August 2009), and steadily became somewhat less active from October 2010 and onwards. In July 2011 the

⁷¹ For Twitter’s API documentation, see: <https://dev.twitter.com/docs>.

⁷² For more info on Twitter’s rate limits and their policies, see: <https://dev.twitter.com/docs/rate-limiting>.

Google group was closed, superseded by Twitter's own efforts to establish a community forum at dev.twitter.com – Twitter's own developer site.

Developer communities are an important aspect of programming in general.⁷³ Most of my informants do not develop software using the Twitter APIs as their main source of income, although many of them have day jobs related to programming and the Web. In many cases, being a third-party developer implies tinkering with the APIs during one's spare time, after working hours. Thus, having other developers to talk to and a place to exchange information, debate and discuss the API, and ask for support and help is essential. As Jeremy, one of my informants, points out:

Every developer community usually gathers in one or more places to interchange knowledge. That interchange is essential to the progress of programming. In this case, as in many others, the Twitter Google Group concentrates much of this activity.

Code in this context becomes a social object, something that moves in and out of the computer, into mailing lists and discussion forums and other discursive spaces. Indeed, software is seldom a solitary effort. Most software can be considered projects in every sense of the word. As projects, software involves numerous actors, including programmers, managers, company CEOs, business partners, venture capitalists, imagined users, test users, discourse, technologies, programming languages, mathematical formulas, and so forth.

According to my informants and the discourse within the Google group, the Twitter third-party community draws together developers with different skill sets and technical knowledge. There are those who have hardly any previous programming experience, as well as developers who consider themselves to be highly literate and knowledgeable in regard to code and coding.

⁷³As has been pointed out especially with regards to free/libre/open-source software (FLOSS), programmers have many ways of gathering (online) to express, share, and circulate ideas about code and coding. As Kelty for instance suggests: 'Geeks live in specific ways in time and space. They are embodied and imagining actors whose affinity for one another is enabled in new ways by the tools and technologies they have such deep affective connections to' (2008:77). In fact, the cornerstone of developing FLOSS projects hinges on a certain idea of community, as collaborating on making the code better is what drives the ideology of both free software and the open source movement. These collaborative efforts and the communities around them in many ways grew out the Internet itself, affording coordination and collaboration through mailing lists, chats, and so forth (see Kelty, 2008: 213-214).

Consider Peter, one of my informants who had been a very active participant in the Twitter developer group (when it still was hosted as a Google group).⁷⁴ In one of our e-mail exchanges, he describes himself as extremely well-versed in programming, with skills in various languages. In his words: ‘my "talent", as people call it, is the fact that I can learn any programming language within three days, and have a proper application ready in seven’. For him, programming is arguably a very important part of his identity, and he spends a lot of his spare time (after school) on developing projects on a hobby basis. At the time of my first interview with him in October 2010, he had just gotten into developing web applications using the Twitter API. Ten months later, in July 2011, he had already developed several apps and clients, and helped many other developers with their apps as well.

Many of my informants seemed to agree that the Twitter API is easy to use and requires little programming experience. As Rob sees it:

Twitter's API is extremely simple to use, developers can access it using almost any language or skill set. Compiled, scripted, even client side JavaScript may be used to interact. This is one of the reasons it's so popular in my opinion, there are thousands of applications available to interact with it.

Because the Twitter APIs are relatively well-documented, easy to use, and because of a thriving developer community, the APIs are accessible to many different people, from complete novices to expert programmers. This is in part due to Twitter’s use of the REST framework and other widely-used interoperable web standards.

While all of my informants have their own unique backgrounds and motivations for getting into programming and for using the Twitter API, many of them were remarkably consistent in the ways they talked about code and the experience of being a programmer. During my interviews with the third-party developers, it became clear how much time, effort, and emotion is being invested in programming practices. Developing software, whether small apps or bigger projects, requires a considerable amount of time spent tinkering, practicing, failing, and succeeding. The emotional

⁷⁴ Peter also represents what can be seen as the typical ‘geeky’ bedroom path to programming. When I asked him how he got into programming, he answered: ‘Fun, work, boredom, filling my time, everything. It all started when I was eight and a magazine I read had a five-page tutorial about HTML. When I was 12, I knew PHP properly, 13 Java and Visual Basic, and by now (17) I don't know [of] a single language which I don't understand (while there are a lot I don't know, I do know that they are essentially all the same)’.

register associated with programming expressed in many of the interviews ranges from hope, happiness, and desire to frustration, anger, and hopelessness. Above all, my informants' spoke of the need for patience and persistence, that coding is all about enjoying solving problems. As Alex, one of the informants, explained when I asked him to elaborate on his programming experience:

Over the years I've read a lot, wrote a lot of bad code and made mistakes. Mistakes are the best things that can happen to a programmer, because it eradicates one more wrong way to do something.

Many of my informants have been programming since they were teenagers. Some have grown up with the rise of the personal computer as an important part of their identity formation. Software means a great deal to these people, as it is part of who they are and what they have become in life. Here, I would like to draw on one particular story that Jacob, one of the informants, told me, as it illustrates how important the computer and programming can become for some, changing the lives of the people it touches. One of the questions that I asked the informants was how they got into programming and what it meant to them. This is Jacob's story:

My school was in a poor neighborhood and didn't have any computers or plans to get them. There was pretty much no hope for someone in my economic strata in the US to see or touch a computer for another 10 years. But then in 1981 when I was 10 years old, Apple donated two computers and paid to have a tutor train the top 10 math students in my school to do basic things like play games and use a word processor. The top two students of the original 10 were taught to program. Me and my friend Jake were the top two. We were taught Logo, Basic, and some COBOL. Both of us went on to get degrees in EE/CS, design microchips, and write software.

Although Jacob did not go on to study computer science at a higher level, but rather electrical engineering, he says that:

Software has always been my hobby. In the early days, electronics and software were more closely linked. Software seemed like a way to play with electronics that didn't cost anything. For a poor kid, that was really important. I couldn't afford to buy more solder ;-)

Passion and/or profit: The discursive regime of API-related programming

As mentioned, while my informants differed in terms of their individual backgrounds and skill levels, as a community of practice they used very similar ways of talking about the Twitter APIs and perceived opportunities. Broadly speaking, many of my

informants used words and terminology that can be said to be reminiscent of what can be labelled a discourse of enterprise, and what might perhaps at first glance seem to be the opposite thereof, namely a discourse reminiscent of a ‘hacker ethic’. However, it became apparent that these are not necessarily opposing discourses, but rather complementary. We can see this dual aspect in Rob’s description of his reasons for using the Twitter API:

My interest in Twitter is purely fascination with their API. I like to dig into things and find out how they work, what's going on behind the scenes, and see what I can do with it that might be of value to others. I'm focused on trend tracking and run promotional material for businesses that want to expand their presence on Twitter without having any technical knowledge.

On the one hand, third-party developers (referring both to some of my informants and the Google group discussions) repurpose a discourse of self-reliance and personal responsibility. Consistent with what can be labelled a neoliberal ideology,⁷⁵ understood as the widespread employment of market concepts in all spheres of society, many third-party developers seem to dream about inventing the next ‘killer app’. On the other hand, developers talk about their love for programming, about passion and a pure interest in tinkering with technology. This, as already alluded to, can be understood in light of what Steven Levy has called the ‘hacker ethic’, as ‘a philosophy of sharing, openness, decentralization, and getting your hands on machines at any cost - to improve the machines and to improve the world’ (2010: ix). Following discussions about the API in the discussion forums of the Google group reveals a similar tendency. For example, in a thread entitled ‘Introduce yourself!’⁷⁶ developers use terminology consistent with both neoliberal/enterprise/entrepreneurial discourse and hacking/tinkering/passion. In this thread, developers introduce

⁷⁵ While the concept of neoliberalism originated from a specific historical context of economic and liberal thought – German post-war liberalism as manifest in the ‘Freiburg school’ on the one hand and the liberalism of the Chicago School on the other – neoliberalism has become somewhat of an all-encompassing catchphrase used by many academics for everything that is seen as negative about the current political and capitalist landscape. For a good overview and outline of the concept of neoliberalism, see Lemke (2001). My use of the concept with regards to the developer community is meant as a way to express the ways in which developers make frequent use of market concepts and economic criteria when talking about their lifeworlds. As Lemke explains, neoliberalism generalises the scope of economics, where ‘social relations and individual behavior is deciphered using economic criteria’ (2001: 8).

⁷⁶ See Twitter Developer Talk: http://groups.google.com/group/twitter-development-talk/browse_thread/thread/d6bd8f0a9b242717 (last accessed February 16, 2012).

themselves by way of saying things like ‘I hope I can make money from it’ on the one hand and ‘Yahoo engineer by day, Twitter hacker by night’ on the other.

These discourses, then, should not necessarily be seen as opposing, but rather as belonging to the same ‘discursive regime’ in the sense of Foucault (1977). As Kitchin and Dodge explain, ‘a discursive regime is a set of interlocking discourses that sustain and reproduce [...] a particular set of sociospatial conditions (2011: 19). Moreover, discourses work together to ‘persuade people to their logic; to believe and act in relation to this logic’ (Kitchin and Dodge, 2011: 262). Rather than representing opposing ways of situating oneself with regards to software, I suggest that the discursive power of APIs both conditions and disciplines. Here, the love for programming and hacking becomes convoluted into the governmentality of neoliberalism, or a kind of ‘enterprise culture’ (du Gay, 1996). As scholars expanding on Foucault’s ideas of governmentality have pointed out, neoliberal discourse in many ways becomes the driving force for passion as well, making work/leisure boundaries less important (see Miller and Rose, 1990).

In this sense, we could say that hacking constitutes but one path to self-fulfilment, where the values of personal responsibility, accentuated by neoliberal discourse, materialise in various practices of hacking and tinkering. That is, love for what one is doing, or what Foucault (2007) called ‘pastoral power’, is used as a driving force for the reinforcement of API programming as a form of ‘entrepreneurial labour’ (Neff et al., 2005). While many of the third-party developers use their spare time to tinker with the Twitter API out of interest, or to help out other developers who ask questions on the mailing list, ultimately this form of passionate or ‘affective labour’ also becomes the key resource in an economy geared towards perpetual innovation. By reinforcing concepts of pleasure and desire in discourses around the Twitter API, we can see ‘how pleasure itself becomes a disciplinary technology’ (Gill and Pratt, 2008: 17). Pleasure and the passion for programming can thus be seen as key techniques of ‘government’ in the sense of directing or guiding the conduct of third-party programmers (see Foucault, 1997).

Third-party developers do not only produce their own applications, nor do they only engage in passionate work, coding for fun. As a community, these developers and programmers also participate in producing the underlying Twitter service itself. As much of the discourse surrounding Twitter suggests, the developer ecosystem has

contributed greatly to the success of Twitter. As Nick puts it in one of his e-mails, ‘I think third party developers were really pretty essential to Twitter’s success’. Similarly, Carl says,

I don’t think Twitter would have taken off without an API. It allows third party developers do things with Twitter and tweets that Twitter as a company did not initially think of.

Carl and Nick aptly describe the double articulation of power manifest in web-based open APIs. On the one hand, as Kitchin and Dodge point out, APIs have lowered the barrier to entry, empowering more people to be creative. As they say, ‘software offers a growing proportion of people with a set of tools to do work in the world’ (Kitchin and Dodge, 2011: 133). On the other hand, APIs have become one of the greatest resources for commercial Web 2.0 platforms, by leveraging on coding practices and developer ecosystems in a fashion similar to the kind of voluntarism seen in FLOSS projects. While the communities of practice surrounding the Twitter API are certainly reminiscent of the kinds of collaborative efforts found in free and open source software development, in the age of the interoperable API, innovation is encouraged on the subservient systems as opposed to on the source code of the core system. Tony, an informant, elaborates on this new form of governance in the following way:

Basically they are taking, what is probably their biggest direct cost, PROGRAMMERS, and dilute that cost amongst a huge base of programmers that not only code for less but also provide the largest source of new ideas for their software. They are making sure they stay in the retail software game as it evolves.

The management style (or protocol) that Tony talks about in many ways reflects what Luc Boltanski and Eve Chiapello have described as ‘the new spirit of capitalism’. Looking at the changes in management discourse and the emergence of the ‘firm as network’, Boltanski and Chiapello suggest that:

[t]he integrated large firm [...] cannot improve its performance in all tasks simultaneously. It must therefore keep in-house only those operations where it possesses a competitive advantage – its core business – and outsource the other operations to subcontractors who are better placed to optimize them. It maintains close and enduring ties with these subcontractors, continually negotiating terms and conditions, and exercising control over production (Boltanski and Chiapello, 2005: 75).

While Boltanski and Chiapello write specifically about new management techniques and the proliferation of outsourcing services, the ways in which APIs organise

participation and coding certainly reads in a similar way. In this sense, Twitter can be seen a networked firm, and its developer ecosystem as the subcontractors. Similar to how Boltanski and Chiapello describe the new spirit of capitalism, Twitter continually negotiates the terms and conditions for what can happen, where and when, and exercises control over software production. However, we should be wary of reading too much exploitation into these relations, as sometimes can be the case with academic discourses around creative, affective, or immaterial labour.⁷⁷ Rather, the dynamics seem to evoke the twinning of freedom and control that underlies the Internet as a whole (see Galloway, 2004; Chun, 2006). As Wendy Chun argues, ‘control and freedom are not opposites but different sides of the same coin’ (2006: 71). Control does not oppress, but rather enables openness. For Galloway, the kind of control realised through protocols is different than one might first think, as it is a ‘type of control based on openness, inclusion, universalism, and flexibility’ (2004: 142). While control is the foundation of both freedom and the Web, Chun also points out that freedom exceeds control (2006: 291).⁷⁸ As much as protocological power regulates or controls, it also produces users’ needs and desires. In the case of the API, we could say that the love of programming is used as a type of control. At the same time and beyond exploitation, the API offers a potential for new subjectivities and socialities. The following section introduces the concept of the quasi-object, in order to consider the productive power of the Twitter APIs as a catalyst of specific collective associations or programmed sociality.

The construction of collectives: APIs as quasi-objects

Application programming interfaces are protocols in every sense of the word. Not only do APIs regulate how data can be exchanged between two software programs,

⁷⁷ Here I am thinking especially of some of the discourse that has emerged around the ways in which the digital domain has mobilised the widespread use of ‘immaterial labour’ as a means of capitalist exploitation. See for instance Hardt and Negri’s ‘Empire’ (2000), or Terranova’s ‘Free Labor’ (2000) and ‘Network Culture’ (2004).

⁷⁸ While a detailed analysis of Chun’s argument is beyond the scope of this chapter, it needs to be pointed out that it is a bit more complicated than it seems. Chun draws upon Jean-Luc Nancy in her conceptualisation of freedom as the condition of possibility. Accordingly, freedom is conceptualised as the beginning. As power, freedom as initiality allows existence to emerge. Because freedom is nothingness, or precedes us, it exceeds control. See pp. 290-297 in Chun’s *Control and Freedom*.

they also direct netspace, code relationships, and connect life forms (Galloway, 2004). I argue that APIs bring actors together in particular relations that point to the regulatory powers of protocols as important diagrams for organising and governing participation on social networking sites. It is not that APIs are simply there for users and software developers to gather around. Rather, following the French philosopher Serres, APIs can be understood as *quasi-objects* that play an active role in configuring social relations. According to Serres, the quasi-object makes the collective (1982: 225).

A huge inspiration to the development of Latour's relational Actor-Network Theory, Serres sought to describe the ways in which social bonds are never only between subjects, but always already imply the participation of objects, or rather quasi-objects.⁷⁹ As discussed in Chapter 2, Michel Serres offers the example of a ball to illustrate his point about the productive force of objects. In a rather poetic fashion, Serres elaborates on the capacity of the quasi-object to bend human practices:

Around the ball, the team fluctuates quick as a flame, around it, through it, it keeps a nucleus of organization. The ball is the sun of the system and the force passing among its elements, it is a center that is off-centered, off-side, outstripped [...] The object here is a quasi-object insofar as it remains a quasi-us. It is more a contract than a thing, it is more a matter of the horde than of the world (1995: 87-88).

Serres uses the ball in a rugby game to illustrate how there would be no social gathering, no game, without the ball. It brings the players together in constantly shifting configurations. As the quasi-object passes through social formations, it also forms relations among the participants. The example of the ball has subsequently been taken up by others to explain the nature of collective and individual individuation, most notably in the writings of Brian Massumi and Pierre Lévy. In Massumi's reading of the quasi-object, the ball is the game's catalyst; it attracts and arrays the players and defines their roles in the game (2002: 71-73). Importantly, the quasi-object seen as a kind of contract or agreement also regulates the field of potentiality. Following this line of thinking, we can begin to see how APIs are not just products of social

⁷⁹ The term quasi designates the quasi-object as something more than objects. The quasi-object is not determined by its properties, but through the way it moves about, tying relations together. As Brown points out: 'This is more than a simple object. It is "quasi" object since it is undetermined, its particular qualities are unimportant. Its [*sic*] standing comes from the way it moves as a token. It is this movement that holds together the players' (2004: 394).

formations, but rather constitute the loci of a range of organisational activities.⁸⁰ The API as protocol or contract ‘outlines the playing field for what can happen, and where’ (Galloway, 2004: 167). The question then becomes how we can begin to understand the playing field that the Twitter API regulates. What kinds of relations are assumed to be desirable, and whose interests are emphasised or devalued?

Understanding the playing field of the Twitter API

As contracts, APIs both promise to deliver something and regulate the relations between the parties involved. Every developer who wants to use the API to access and repurpose the data in a third-party application or website needs to conform to a set of rules. According to these rules, Twitter has the right to terminate or shut down any third-party app that does not comply with the rules. Importantly, the rules describe ‘what type of innovation is permitted with the content and information shared on Twitter’.⁸¹ The contract also states that ‘Twitter may update or modify the Twitter API, Rules, and other terms and conditions, including the Display Guidelines, from time to time’. This makes the Twitter API a risky thing, as developers lose important control over their products when they make their applications reliant on the APIs. Importantly, APIs carry with them an assumption about what constitutes innovation and desirable coding practices. APIs thus occupy an interesting position of double articulation, involving control and freedom, threat and opportunity simultaneously.

APIs not only open up access to data and software functionalities, but also a field of possibility for capital, creativity, and coding. As Carl, an informant, points out: ‘our company would have a hard time existing without the Twitter API’. Jacob, another informant, pointedly suggests: ‘my business relies on them’. Indeed, for many of the developers, the Twitter APIs constitute the condition of possibility for their applications and businesses. In this sense, the APIs act as catalysts, not merely for apps and businesses, but also for becoming a programmer. If the 1990s, as Kitchin and Dodge suggest, underwent a certain democratisation of technology where software contributed to a burgeoning of ‘back bedroom’ creativity through desktop

⁸⁰ Harris (2005) points out that this is precisely the approach ANT would take, to place the notion of quasi-objects at the heart of understanding social organisations (p. 173).

⁸¹ See ‘Developers Rules of the Road’: <https://dev.twitter.com/terms/api-terms> (accessed 15 February, 2012).

design and home recording setups (2011: 121), the last decade of burgeoning APIs has dramatically opened up new possibilities for creativity through ‘mashups’ and apps. Tony talks about this, when I asked him how he perceived the impact of APIs:

I believe APIs are a great thing because they have let people with ideas, come one step closer to being able to put those ideas to work, by not having to know too much about programming.

Of course, as we have seen, APIs are not simply objects that beget the third-party programmer. While these coded systems and programmable interfaces to web services allow for a much greater access to data, APIs need to be seen as designed spaces, like any other user interface. As designed spaces, APIs are suggestive of certain kinds of actions, and afford certain kinds of coding practices. In this sense, APIs also close the very same possibility space, by acting as a gatekeeper and regulator. APIs demarcate function, they control access to data, determine the boundaries for innovation, and define what can and cannot be done. In my interview with informant Jack, he elaborates on the dual existence of control and freedom, opportunity and threat:

There's been a huge wave of "just open it up and see what happens" that we're just at the beginning of understanding. The implications of which will shudder some businesses, while allowing some to flourish. What the underlying API supports, or doesn't, indeed is defining social media as we know it. APIs define what we can build, policy-wise, as well as technically, and subsequently the products we build/use/consume, which in turn obviously affect culture and socialization in general.

Like Serres’ quasi-objects that have the capacity to change and affect the relations between actors as they pass through and circulate across different assemblages, the Twitter API stabilises some relations while destabilising others. Indeed, as Jack suggests, what the underlying API does or does not support defines social media as we know it. Who is affected and in what ways depends on the positions of the actors involved. To invoke the example of the ball and the game again: while the API is indeed the focus of every player on the field, the ‘relationship between the players is defined by how they position themselves with regard to ball’ (Brown, 2004: 394). Some have a better chance at winning, while others are at risk of losing.

If we take the ball analogy one step further, we might begin to see the many ways in which the ball signifies both opportunity and threat, not as a thing that just lies there on the ground, but as something that in conjunction with the player has the capacity to induce action and movement. In a similar vein, we could say that the Twitter API

does not do anything in and of itself. Like the ball, the API signifies and prompts action. The API is there to be used. Like the ball, the API seems to suggest ‘play with me!’ ‘Be creative!’ At the same time, ‘play’ through the API is rule-bound. The API documentation and Developer Rules provide the structure of the game, regulating what can and cannot be done. As in the ball analogy, there is always the risk of being tackled. In Serres words, ‘with the ball, we are all possible victims; we all expose ourselves to this danger and we escape it’ (1982: 227).

How can we understand this in terms of the playing field created through the Twitter API, and what are its risky opportunities? I will elaborate on these questions in the next section, where I discuss a specific case of controversy in terms of Twitter’s restrictions and regulation of their APIs, and the effects on the third-party developer ecosystem that played out during March 2011.

Governing innovation

In this section, I turn to a more specific case in which the Twitter API underpins freedom and control, opportunity and risk. I argue that the ways in which the Twitter API seems to concatenate relations – that is, define the playing grounds for what can happen and where it can happen – hinge on parameters of governance that have the regulation of innovation as its goal. As de Souza and Redmiles point out, the protocological aspects of an API imply a certain division of labour, rules, and conventions that need to be attended to if we are to understand its powers beyond the obvious design and usability issues that have been the focus of previous API research (2009: 447). The specific rules and conventions underlying an API constitute a necessary framework for how user data can be accessed in terms of privacy concerns. As long as the participants are content, the governmental mode of managing the actors involved within the playing field remains more or less unproblematic. However, when the regulatory nature of these conventions becomes a matter of contestation, the divisions of which de Souza and Redmiles speak become particularly visible. Employing sites of ruptures as a means of getting closer to otherwise invisible objects or phenomena is a well-known strategy within the philosophy and sociology of technology (see Heidegger, 1977; Latour, 2005). Mackenzie seems to suggest that one way of understanding software and its transformative capacities is to look for the ways in which software becomes visible as sites of contestation (2006: 3). Mackenzie

identifies some of the differing ways in which software becomes visible through specific debates and discourses that turn it into a highly invested object, as for example was the case with Apache webserver or the Y2K bug (ibid.).

In a similar vein, I will now turn to the governance of innovation implicated by the Twitter APIs. In doing so, the following section takes a closer look at one particularly contested and debated case of API controversy. The case of Twitter directly speaking out and ‘prohibiting’ development of certain Twitter clients became a much-cited and debated issue, both among developers themselves as well as within the broader social media industry (see Arthur, 2011; Siegler, 2011).

It’s not personal, it’s business: Twitter ‘prohibiting’ third-party clients

While Twitter has openly encouraged third-party development from the very start and used considerable resources to build stable and consistent APIs, eventually there came a time when too much innovation turned into a potential threat on the part of the core service. That is, too many similar systems were suddenly on the market, competing with each other for the same users. For Twitter, the ecosystem that it had so eagerly supported became a potential threat; the plethora of subsystems that had been generated using the Twitter APIs suddenly threatened to make the core service obsolete. As a result, Twitter decided to tell the ecosystem of developers to stop building new third-party clients. On March 11, 2011 platform manager Ryan Sarver, acting as a spokesperson for Twitter, posted a message on the Twitter Developer Talk list, where he urged developers to stop making new Twitter clients, and claimed developers were confusing the user experience of Twitter.⁸²

The message was that too many developers were apparently doing the opposite of innovation, namely making more of the same, merely producing replicas of Twitter itself. As Sarver put it, ‘consumers continue to be confused by the different ways that a fractured landscape of third-party Twitter clients display tweets and let users interact with core Twitter functions’. He argued that users should have the same

⁸² Ryan Sarver’s post was titled ‘Consistency and ecosystem opportunities’ and was posted March 11, 2011. The message quickly turned into a discussion thread where it received 92 replies within the following three weeks. See http://groups.google.com/group/twitter-development-talk/browse_thread/thread/c82cd59c7a87216a. The quoted comments below are taken from this thread.

experience of Twitter regardless of which clients or applications they use. Looking at this from the perspective of the previously-introduced game analogy, one sees that the playing field appeared to have gotten out of hand; the rules were not as apparent anymore, and too many players had begun playing the game on their own terms. The rules needed to be straightened out; it needed to be made clear who had the power to define the rules and conventions of the playing field.

While the game welcomed every player at the start, the time had come to take out the team. Who were the valuable players, and who seemed to do more harm than good? Ryan Sarver's message to Twitter's developer ecosystem provided a clear indication:

Developers ask us if they should build client apps that mimic or reproduce the mainstream Twitter consumer client experience. The answer is no. If you are an existing developer of client apps, you can continue to serve your user base, but we will be holding you to high standards to ensure you do not violate users' privacy, that you provide consistency in the user experience, and that you rigorously adhere to all areas of our Terms of Service.

This quote clearly shows the risky territory created by the APIs. Developers will be held to 'high standards' so as not to violate the Twitter terms of service (TOS) or users' privacy. However, the boundaries are unclear. When is something considered a breach of privacy or a violation of the TOS? When is a specific mashup or app considered to be 'good enough', a true innovation? The reactions following Twitter's prohibition of third-party clients illustrate some of the feelings, intensity and significations passing through code like any other cultural object (Mackenzie, 2006: 5). Immediately, someone ironically posted the following message in the discussion thread: 'Wow. Thanks for getting so many people interested in Twitter. Now get lost. This is appalling.' Another developer provided an apt summary of what now seemed to be at stake for the developer community: 'All third party Twitter developers, no matter what they make, are now walking on eggshells, constantly at risk of offending Twitter's ideas of how users should interact with Twitter'. Yet another developer's reaction shows the potential risk that API changes have for software developers: 'You've just scared the bejesus out of me because I don't know if I'm suddenly verboten or not. Five months of work shot to hell?'

The case of the Twitter client prohibition illustrates the risk of programming using an API. As Schroeder observes in a blog post on Mashable:

Is it too dangerous to build an application on an API you can't control? Whether it's Twitter or Facebook or Digg or any other web service that provides a public API, creators of third party applications are always at risk that their efforts will simply be erased by some unpredictable move on the part of the company that controls the API (2009).

While the tenet of a good API is stability, APIs do change. Sometimes it is the API itself, as in the actual calls or methods for fetching data. When this happens, apps that make use of these methods instantaneously die off if they do not make the necessary amendments. This means that programming using the API of a web service requires ongoing attention on behalf of the third-party developer. While some of my informants thought that the possibility of change was in fact positive, as it made them improve or update the apps to keep up, others felt rather frustrated by this level of insecurity. Developers and API providers alike see the APIs as contracts. Just as the providers expect developers to adhere to the rules of the road and the TOS, the developers expect the APIs to remain stable. Changed APIs imply significant extra cost for third-party programmers. In some cases, changed APIs imply a complete waste of time, and perhaps more dramatically the loss of a business. This is exactly what happened to one of my informants, Jacob. When I asked him about his interactions with the Twitter API, he replied:

I'm no longer interested in contributing anything to Twitter's API. Their hostile stance toward developers like me has been very discouraging, not to mention costly -- they killed my business, it has cost me many thousands of dollars.

Jacob, who had been working on a Twitter client for a long time, said the Sarver incident finally killed his business. In his opinion, Twitter had gradually become more hostile towards their once-profitable ecosystem. They had made it more difficult to interact with the APIs, both in technical and legal terms. In relation to the Sarver case, Jacob explained:

This is the third major change in authentication system in less than two years. Each requires a significant amount of engineering. Each frustrates users. And with each Twitter offers their own in-house app, free and without headaches.

While most of my informants seemed to have a positive attitude towards Twitter, holding the company up as a prime example of a successful and sympathetic start-up, there was also a sense that they, as a community of developers, had somehow been let down. The community thrived in the days of the Google group, but this changed once Twitter itself hosted the forum.

Peter, who was a very active member of the Google group community, thinks, ‘the community we had has simply died’. As he elaborates:

The "original" community is now simply e-mailing questions directly to Twitter, or have [sic] stopped developing. Only a very small group of them still uses dev.twitter.com, knowing that they don't get an answer anyway.

Jacob echoes the sense of decline of the community:

I think the group was much more vital in previous years. It seems to me now it's less of a community now and more often a PR channel and dev support forum. In previous years there were many more community contributions from third parties. And those contributions were encouraged much more. Now there are few and Twitter is openly discouraging to many types of third part development

While we can sense some level of resentment in this quote, Jacob is also quick to add:

That said, the engineers within Twitter have always been very nice to me personally. I've had the opportunity to meet many of them in person and they're really great guys. This bipolar nature indicates that the hostility is coming from elsewhere, perhaps management? Or maybe the money guys? I don't know.

My intention is of course not to assign blame to any one party for something that arguably is part of the game. Rather, it is to point at the game itself, its players (read various participants), positions (in relation to the API), desires, and motives. The playing field controlled by the protocols of the Twitter API also enable and make the game possible in the first place. On the one hand, the APIs have opened up new possibilities for creativity and innovation. On the other hand, the same freedom comes with a loss of control.

Consistent with neoliberal logic, third-party developers have to be highly flexible subjects, willing to take the risk that this kind of cultural work entails. However, as the notion of APIs as ‘objects of intense feeling’ attests to, the costs – both personal and professional – can sometimes be too high. For someone like Jacob, the implications might simply mean losing the game, because of the ways in which the quasi-object of the API moves in unidentified ways. While APIs have arguably become the basic building block of the social Web, cases like the Twitter third-party client prohibition shows that ‘sometimes applications might be building their services on a foundation of sand’ (Higginbotham, 2011).

Conclusion

As we have seen, APIs shape, control, and enable practices of sharing, transmission, and innovation in multiple ways. On the one hand, APIs provide the condition of possibility for what is increasingly talked about in terms of ‘big data’, the proliferation of data managed and produced on the Web today. APIs make it possible for different actors to share and use this data, enabling new services and software to come into existence. For many of the third-party developers I interviewed, APIs are seen as a new powerful resource, bound up in and contributing to new opportunities and desires. Whether these come in the form of business opportunities, by way of monetising the data provided through the APIs, the opportunity to enhance one’s programming skills or even getting into programming altogether, or just new and fun way of practising one’s love for programming, the Twitter APIs are objects of lived experience. APIs thereby have very real material effects on end users.

On the other hand, APIs regulate and restrict the same flow of data and information that they enable. APIs set the limits for what can be shared, and in which ways. Through techniques and procedures such as rate limiting and highly-controlled access points, APIs act as highly-powerful gatekeepers of data. Importantly, APIs are capable of responding to, adjusting, and modulating various changes within the assemblages in which they participate.

As I have argued, these convoluted capacities of the API to make things happen, to organise sociality in certain ways could usefully be understood through Serres’ notion of the quasi-object. As such, the Twitter API can be understood as a catalyst for the formation of collectives and individual being. Not only do the Twitter APIs condition the existence of the developer ecosystem; APIs also constitute an important ‘communicative mechanism’ (de Souza and Redmiles, 2009). The Twitter API, as we have seen, gathers multiple actors into communities of practice and ‘discursive regimes’ involving a ‘variety of vested interests’ (Kitchin and Dodge, 2011: 251). While some of the online interview data was collected only a couple of months after the Sarver incident, which arguably influenced the sentiments expressed by the developers towards Twitter at that time, my findings clearly show that APIs are never only neutral objects.

APIs should be understood as ‘objects of intense feelings’. They are invested with various forms of contestation and identification, hopes and disappointments. Their protocological power stems from their capacity to define the playing field of innovation and to gather actors in various constellations of collaboration and controversy. As I alluded to with the coupling of control and freedom, power through the API needs to be seen as emanating both from the techniques and procedures that it prescribes, and from the developers themselves. As a catalytic device, the Twitter API activates certain impulses and relations. As we have seen, these relational impulses include the divisions of labour (i.e. Twitter core service and apps as subsystems, Twitter engineers and third-party developers), the construction of rules and negotiations (i.e. continually updating terms of service and API usage rules), as well as the promotion of creativity, hopes, fears, risk, and pleasure.

In this chapter I have sought to illuminate APIs as sociotechnical constructs that have the power to install both opportunity and threat. APIs, it was argued, are not simply ‘specifications and protocols that determine relations between software and software’ (Cramer and Fuller, 2008: 149), but can also be analysed as quasi-objects, protocols that shape the relations between multiple actors, including humans and nonhumans.

Chapter 9. Conclusion: Turning the tables on ‘digital humanities’

The main objective of this dissertation has been to examine the ways in which software shapes sociality in the context of social networking sites. The aim has been to use case studies to illustrate how an understanding of social life online benefits from an analysis of the ways in which non-human actors, especially software, participate in the activities and interactions between the entities (both human and nonhuman) concerned. My main argument is that software fosters and supports what I call ‘programmed sociality’, understood as the ways in which software codes, assembles, and organises ways of relating to oneself and others. This is not to say that user interactions and practices on social networking sites are pre-programmed and determined by non-human actors. Rather, the concept of programming allows us to consider sociality as an on-going process of assembling and reconfigurations that involve human and non-human actors. In this dissertation, ‘programmed’ has not merely referred to the materiality of software, but also to what computer pioneer John von Neumann called ‘to program’: that is, to ‘assemble’ and ‘organize’ (Grier, 1996: 52). This has allowed me to conceptualise software as both material and practice, something that has been constructed and designed in certain ways, and a procedure for assembling and organising that to which it pertains. I have been guided by Foucault’s understanding of power as a productive force, and have used the concept of power as a primary analytical framework in which to understand the impact of software in shaping sociality in social media. My goal in this dissertation has thus been to show and illuminate some of the ways in which the productive power of software plays out in the context of Facebook and Twitter.

I have shown how software governs the conditions of the intelligible and sensible by propagating a certain social order of continued participation. In chapters 5 and 6, I demonstrated how the protocols and algorithms of Facebook function as important steering and selection mechanisms for what users are allowed to see and to what their attention is directed. The studies on attention and visibility on Facebook offered a way to address the question of how software organises and distributes the sensible.

Software permeates social relations by playing an active role in the individuation of collective associations. In chapters 7 and 8, I showed how software not only supports

sociality, but is in fact what makes sociality possible in the first place. Although quite different in scope, the studies on friendship formation and API interaction show how software needs to be seen as an actor and active participant in shaping social formations and groups.

In this concluding chapter, I will expand on the notion of programmed sociality by relating my arguments and findings to the broader picture of networked society and algorithmic culture. I will begin by elaborating on the main contributions of this dissertation, before discussing some of the wider implications of my research.

Major contributions

This dissertation provides several empirical, methodological, and theoretical contributions to media and software studies. First, I have shown how software can be analysed as something that shapes the conditions for sociality on social networking sites. Despite the fact that software studies is a relatively new field, there are as many ways of pursuing the study of software as there are ways of conceptualising software. Secondly, this dissertation has developed what I have called a *technographic* approach to the study of software, which can be used as a tool for analysis. Finally I have shown how the subject-software continuums examined in the previous four chapters are examples of what I call ‘programmed sociality’. In the following section, I will expand on each of these points.

Towards a software-sensitive perspective to media and communication

In regard to the field of media and communication studies, this research contributes to the understanding of social networking sites by invoking a software-sensitive analytics. This dissertation provides an example of how we might understand and critically approach important issues within media and communication studies, including attention and visibility, by questioning the material-discursive conditions of these media as software. Such a perspective can be contrasted with a more traditional manner of analysis: that of studying users. A software-sensitive analytic approach does not mean that users are disregarded; it simply implies another perspective on the user – a perspective that I believe has been overlooked in media studies.

I have shown how applying a software-studies perspective on social networking sites brings about a new understanding of the co-mingling of actors in these spaces, and how an analysis of sociality in networked media benefits from taking software actors into account. While I have focused on aspects such as attention, visibility, friendship, and developer community and practice, the usefulness of taking a software-studies perspective extends well beyond these issues.

One could apply a software-sensitive perspective to almost any topic or issue of interest within (new) media studies; therefore, my approach has applications beyond my focus on social media. My research has contributed to a new understanding of the issues I have studied, for example of how attention is arranged in certain ways, how visibility becomes an algorithmic strategy etc., and I have shown how taking software seriously opens up for understanding traditional media studies topics in new ways.

The question is whether social life on the Web can be studied without taking into account the plethora of software actors that contribute making sociality meaningful. I believe it cannot. One of the critical tasks for media scholars is thus to rethink some of the traditional topics within media and communication research in light of algorithmic and coded interventions. It is necessary to consider the regulatory spaces entailed by the growing use of algorithms in everything from agenda-setting, gatekeeping, editorial processes, rhetoric, advertising, and political participation and the public sphere, just to name a few areas of interest to media research. This dissertation offers one possible way of synthesising media studies in a manner that has its point of departure in the power of software.

Towards a technographic approach to the study of software

A second major contribution of this dissertation is to offer a version of software studies that hinges on what I have called a technographic approach. As I discussed in Chapter 4 on methods, a technographic approach is a way of reading and describing technology – in this case software – in a manner similar to that in which an ethnographer would describe and interpret culture. Replacing ‘ethno’ with ‘techno’ signifies a shift in focus, from people’s lifeworlds to the things that software itself can be said to be suggestive of. Technography provides a way to treat software elements and processes akin to other types of cultural texts, meaning that they can be

approached as assemblages of various ways of sensing, doing and knowing. This way of engaging in software studies implies an attentiveness towards the various traces and elements of software clustering around social networking sites, including blog posts, partial descriptions of the algorithmic logics, tech talks explaining parts of the engineering that goes into these systems, available technical specifications, media coverage etc. This, I believe, provides a useful approach to software and software-mediated processes that exists somewhere in between a study of the interface alone, and reading the source code.

My contribution to software studies has been to offer a detailed analysis of the stuff of software, to account for the important role that protocological and algorithmic actors and processes play in the context of social media - an area of research that has largely been missing in writings on software studies thus far (with some notable exceptions of course, see Chapter 2). By focusing on digital objects and ensembles such as the Open Graph protocol, the EdgeRank algorithm and other algorithmic and coded actors of the Facebook platform, in addition to the Twitter APIs, this dissertation has opened up a space for a kind of software studies that seeks engage with the micropolitics of power through software.

Towards an understanding of 'programmed sociality'

A third major contribution that this dissertation makes to media and software studies is to show how software signifies and is suggestive of things in the context of networked environments, in terms of producing the conditions for the sensible and intelligible. Software acts not only as a mediating force in social formations online, but also as an active constituent of the practices and relationships that these formations entail. The programmatic techniques, procedures, and mechanisms at play in social networking sites not only make sociality possible, but also govern how and in what ways collective associations form. By considering the notion of 'program' in the sense of von Neumann – as a way of arranging and organising – sociality in networked environments can be understood as being organised (read: programmed) by various coded systems and processes. I believe the term 'programmed' offers a useful way to articulate the type of traversal temporalities embedded in the various algorithmic processes and data-driven logics underlying social networking sites. In

particular, the term programmed captures the ways in which these computational processes engage in the formation of a future based on past data.⁸³

In this sense, programmed sociality exists at the intersection of various temporalities, where the productive capacity of software both constitutes and is constituted by relations to past, present, and future. Chapters 5, 6, and 7 demonstrated how the protocols, algorithms, and other programmatic techniques of Facebook work to ‘make sense’ of past conduct in order to generate new possible futures. Programmed sociality points to the productiveness of software, as evidenced by the exercise of power. As Foucault suggests about the essential nature of power:

In effect, what defines a relationship of power is that it is a mode of action which does not act directly and immediately on others. Instead, it acts upon their actions: an action upon an action, on existing actions or on those which may arise in the present or the future (Foucault, 1982: 789).

In Chapter 5 we saw this dynamic of power, understood as acting upon actions, in the ways in which attention is managed and organised to enable certain results. By describing the infrastructural transformations of the Facebook platform, the argument was made that Facebook does not merely enable certain actions and connections to take place. Rather, the software acts upon the actions that it merely claims to facilitate in order to create and capture users’ attention. Importantly, this process of creating certain modes of attention is not just confined to the competition of ‘eyeballs’ or ‘clicks’, but designed to generate one particular kind of attention: participation.

In Chapter 6, we saw how this data-intensive environment plays out in the context of Facebook’s News Feed. The heavily-personalised News Feed operated by the EdgeRank algorithm creates a regime of visibility that, like the management of attention, hinges on the governance of participatory subjects. Visibility on Facebook is not something ubiquitous, but rather a scarce resource where there are no guarantees of becoming visible as part of the News Feeds of one’s friends. In other words, the fact that one user sees content of a certain type does not mean that everyone else also sees it. In fact, as I demonstrated via an experiment of reverse engineering, becoming visible on the News Feed is not only highly regulated, but is in

⁸³ The notion of ‘programmed sociality’ is inspired by Wendy Hui Kyong Chun’s book *Programmed visions: Software and memory* (2011). As she writes: ‘New media proliferates “programmed visions,” which seek to shape and predict—indeed embody—a future based on past data’ (2011: xii).

many respects also an exclusive affair, a fact that corresponds quite well to recent numbers reported by Facebook itself.⁸⁴

Applying the notion of programmed sociality to Facebook's News Feed allows us to consider the micropolitics of power that are embedded within such programmatically-designed media spaces. Ultimately, protocols and algorithms need to be understood as associative devices that make certain decisions about which relations to forge, and whom and what to connect. What is important to emphasise, is the fact that the connections forged by software have a politics. The linkages are not formed in an arbitrary manner, but rather designate a strategic arrangement aimed at governing the conduct of individuals or groups in certain manners.

Sociality however, is not set or organised once and for all. Rather, the materiality of software and the modes of being together as organised by the software are continuously in flux. Algorithms are ontogenetic simply because the problems that need solutions continually change. While the algorithms at play in Facebook continue to calculate interestingness (the problem that needs to be solved), the company's definition of interestingness continually changes as new goals are introduced and contexts change. For example, when Facebook introduced a set of new apps at the *f8* in 2011 aimed at promoting activities such as listening to music or reading newspaper articles, the algorithms were tweaked to make those Facebook users who were actively using these new apps more prominent on their friends' News Feeds. This means that users who were using these apps were becoming more visible than others. By marshalling what is visible and invisible, algorithms have the power to control the boundaries for what and who becomes recognised, and perhaps more profound – who or what is considered insignificant or simply forgotten. This kind of subject-software dynamic shows that there is a need to be attentive to the contexts, workings, and implications of algorithms, especially as contexts change in such rapid and often unpredictable ways.

The notion of programmed sociality helps to illuminate not only how sociality itself has become the primary object of social networking software, but also how sociality is programmatically 'transformed, redistributed and deployed' (Mackenzie 2006: 173).

⁸⁴ According to Josh Constine of TechCrunch, 'your average Facebook post only reaches 12% of your friends'. <http://techcrunch.com/2012/02/29/facebook-post-reach-16-friends>

In Chapter 7, I addressed this notion explicitly in my analysis of the friendship assemblage formed on Facebook. Users do not only forge connections with ‘friends’ via online platforms; the platforms themselves also contribute to the creation of these social connections.

There are as many ways of programming sociality as there are ways of forging and managing connections. Friendships on Facebook are but one form of programmed sociality, albeit one that provides a very useful example of the work that software does in managing and arranging relationships. The notion of programmed sociality does not only imply a linear processes whereby relations are concatenated in a predetermined fashion. The software actors explored in Chapter 7 do not determine friendships, as people who become friends on Facebook already have an existing offline relation in most cases. Rather, software, by virtue of its active role in shaping specific conditions for being together online, performs work of meaningfulness that configures collective associations in new ways. Whether we look at friendship relations or other forms of sociality in computational media, what is important to take into account are the ways in which actors are continuously connected and disconnected as part of a subject-software continuum.

As was shown in the previous chapter on APIs, taking into account the multiplicity of actors that participate in the construction of sociality complicates the picture of how software works in the world. APIs do nothing by themselves, but once they become part of programming practices, developer guidelines, rules of conduct, and part of discourse and community formations, they become invested with the power to afford, open up, but also constrain new possibilities and determinations.

Ultimately, I believe much is to be gained from understanding the role software plays in social networking sites through an examination of its productive power. This dissertation shows how software introduces new forms of governance by assembling and organising conditions for the sensible and intelligible on social networking sites, thereby altering how participation in these media spaces is regulated. In doing so, the case is made for understanding social networking sites as an example of programmed sociality, and that such an understanding requires an analysis of the various components of how sociality is translated into code and how the resulting software reshapes social life. Much of course remains to be said and done in terms of working through the politics and power of software in our contemporary information

ecosystem. In the final section of this concluding chapter, I will therefore turn to what may lie ahead, thinking through some of the possible future directions of social media and software studies, and provide some suggestions for further research.

Thinking ahead: Social media and software studies

In the three years during which I have undertaken this PhD research, software studies has emerged as an interdisciplinary field in its own right, using methods and approaches from the humanities and social sciences as a means to study software and its practices. When I wrote my initial project proposal in early 2008, several efforts had been put in place to establish the field in a more formal manner. While it is still a topic on the margins of most traditional media and communication departments, software is increasingly being ‘recognized as an object of study and area of thinking amongst scholars and disciplines that have not historically “owned” software’ (Fuller, 2008: 2). Now that software studies has undergone a phase of formalisation, giving rise to academic courses and job calls for tenure-track positions, the establishment of its own academic journal, an MIT press book series and conference tracks, the question remains: where to go from here? Now that more and more media scholars are seeing the value and importance of taking a software perspective, what does this imply, and how should research on software from a humanities perspective proceed?

In many respects, the year of this dissertation’s publication – 2012 – marks an important year for software studies. One increasingly sees matters of code and software being debated and discussed as part of public discourse. There have been many media reports declaring 2012 the year of code, with numerous new online services emerging that offer free crash courses in how to program. The *New York Times* sees a ‘blooming interest in programming’, reporting that there is ‘a surge in learning the language of the Internet’ (Wortham, 2012), while the *Guardian* devoted its entire weekend edition on March 31st 2012 to issues of code. If the emergence of Web 2.0 and social media was generally accompanied by a celebratory polemic in terms of opening up the possibilities for ‘ordinary’ users to become media producers, we now see the contours of a discourse that holds that it is not enough for users to merely generate web *content*.

As life increasingly becomes permeated by code and algorithms are put at the centre of everything from financial trade to information retrieval and social networks, the question is whether we can afford *not* to take software seriously. As journalist John Naughton (2012) puts it in one of the *Guardian* special issue articles:

So something's happening: there's a sense of tectonic plates shifting [...] The biggest justification for change is not economic but moral. It is that if we don't act now we will be short-changing our children. They live in a world that is shaped by physics, chemistry, biology and history, and so we – rightly – want them to understand these things. But their world will be also shaped and configured by networked computing and if they don't have a deeper understanding of this stuff then they will effectively be intellectually crippled. They will grow up as passive consumers of closed devices and services, leading lives that are increasingly circumscribed by technologies created by elites working for huge corporations such as Google, Facebook and the like. We will, in effect, be breeding generations of hamsters for the glittering wheels of cages built by Mark Zuckerberg and his kind.

Is that what we want? Of course not. So let's get on with it.

The dystopic tone aside, the journalist is right about the need to discuss and understand the impact of networked computing. While the importance of code and the need for code literacy can hardly be overstated, we do not have to wait for another generation to learn how to code before we can get off these so-called 'glittering wheels of cages'. We do not even have to reinvent the wheel in order to engage in a critical study of software. As I have shown, a technographic approach, using well-known methods from the humanities and social sciences, provides a fertile ground on which to address the power of software. Mixing an analysis of new technology with the 'old' theories already familiar to many media researchers not only offers a way to understand the unfamiliar through the familiar, but also serves to rejuvenate 'outdated' theory by revisiting it in terms of the new.

Just to take one example, it may be argued that Foucault's notion of panopticism is a somewhat overused theoretical framework for the analysis of contemporary technology. It has however been very influential. In fact, revisiting the idea of the technical and architectural organisation of power as proposed in the writings of Foucault has provided me the tools with which to understand Facebook's algorithmic logic.

If we are witnessing an increased interest in the *digital humanities*, understood as ways of using computing (including tools such as data visualisation and data mining

of large data sets) to understand traditional disciplines in the humanities, let this dissertation be a reminder that the opposite is needed as well. That is, a way of using the tools of the humanities and social sciences to understand the micropolitics of power entailed by specific software objects and their possible implications. While the digital humanities and the debates surrounding ‘big data’ in many ways bring together social media research with a focus on software studies, we must be wary of being seduced by the promising aspects of new software techniques – yet again.

I find there to be an obvious disconnection between the contemporary calls for more coding skills on the one hand, and the celebratory discourse surrounding data visualisation techniques on the other. By celebrating the promises of visualisation tools such as Gephi too enthusiastically and urging students to become more literate in statistics and network analysis, there is a risk that we will merely educate a new generation of software users, rather than of software critics. While there can be no denying the significant power and potential of big data (the quantity of information produced by people, things, and their interactions), its value derives not from the data itself, but from the ways in which it has been brought together into new forms of meaningfulness by the associational infrastructure of the respective software systems.

While I hope to have provided valuable perspectives on the politics and power of algorithms in contemporary social media, much work remains to be done. The question is not only how to make sense of the increasing amount of data generated by our online interactions, and to ponder how best to visualise these in information graphics. Rather, what becomes imperative is to question the politics behind the software’s own principles of visualisation. Facebook’s News Feed constitutes but one such important site where the software becomes visible by producing the conditions through which data appears and is brought together into new forms of meaningfulness. My research has provided insight into the workings of one of the most important web platforms today, thereby highlighting the need to question the principles behind the presentation of data as part of our most-used media sites.

What I have addressed in this dissertation is not the final situation. Ways of being together and forming attachments online will continue to be programmed by the associational infrastructure of software systems. Media and software studies must not only address the role algorithms play in our media ecosystem, but also the roles we want them to play. Where is the influence of algorithms felt, and what principles of

connections do they forge? What does it mean for a democracy to have delegated the task of making public opinions seen and heard to programmable devices? How do these architectural and algorithmic technologies affect social media content and public discourse more generally?

The bandwagon of digital humanities, big data, and data visualisation will need to be counteracted by rigorous criticism that works out the commercial underpinnings, new hierarchies, and rules of participation that are occurring in and through networked computing. A ‘digital humanities in reverse’ is required in order to ask critical questions pertaining to the proliferation of data and the ways in which the data is being ‘tamed’ by mathematical inference and seductive graphics. If data, as Alexander Galloway (2011) argues, have no necessary visual form, then there is a need for future research to develop a critical understanding of these proliferating algorithmic ‘fabrication’ of forms. How exactly is data being put into new forms of meaningfulness?

Not only is there a need to counteract the (renewed) belief in the use of software techniques to make sense of the world, but also to critique the algorithms doing the actual work of *informing* the data. One of the reasons algorithms are so powerful is that they have the capacity to produce new realities, without having to give much consideration to the particular source and context of the data on which they act. An important task for media studies is therefore to address not only how users generate content, but also how algorithms create certain stories and narratives by putting these data into new forms. Algorithms in this sense need to be addressed as *meaning engineers*, as opposed to merely being viewed as abstract tools created by engineers. This capacity of algorithms to tell stories turns them into self-evident objects for humanistic research.

The burgeoning presence of APIs on the Web presents another point of synthesis where social media research and software studies meet. This dissertation has shown how APIs imply the confrontations of different actors, and how actors transform and reconfigures the social media ecology. My research into the third-party developer community of Twitter provides a novel perspective on the meaning-making capacities of software, in terms of its effect on programmers’ lifeworlds. Much has been written about the social media user, as in the many accounts of the so-called ‘produser’. My research has contributed an understanding of the ‘other’ produser – the API

user/programmer. Thus, I have highlighted the importance of seeing current media practices not only in terms of how the end user ‘makes sense’ of the data presented on the graphical user interface, but also how repurposing software in order to access and remix the data constitutes an important aspect of media practice today. More research is needed into the software cultures that are involved, especially in times where mobile and tablet apps are becoming an important part of our media ecology.

While APIs open up interesting new possibilities for social media research in terms of collecting user data, they are also currently at risk of being treated as just another convenient software tool. As it stands, APIs are *used*, as opposed to critically scrutinized as powerful managers of contingent relations and specific flows of communication. In order to countervail these tendencies, further research should examine the commercial imperatives of social media APIs and how they connect relations. The question, then, is not what we can use the software to do, but what the software *does to* whatever it is being used to do.

Ultimately, I think we are at an interesting moment in the history of social networks and social media, in which assumptions about participation and sociality are becoming embedded in software cultures that are only beginning to be critiqued.

References

- Agamben, G. (1993). *The coming community*. Minneapolis, Minn.: University of Minnesota Press.
- Allan, G. (1989). *Friendship: Developing a sociological perspective*. New York: Harvester Wheatsheaf.
- Ammirati, S. (2007). Twitter's open platform advantage. *ReadWriteWeb*. Retrieved February 6, 2012, from http://www.readwriteweb.com/archives/twitter_open_platform_advantage.php
- Anderson, B. (2010). Preemption, precaution, preparedness: Anticipatory action and future geographies. *Progress in Human Geography*, 34 (6), 777-798.
- Ansell-Pearson, K. (1997). *Deleuze and philosophy: The difference engineer*. London: Routledge.
- Appadurai, A. (1986). *The social life of things: Commodities in cultural perspective*. Cambridge: Cambridge University Press.
- Aristotele. (2004). *Nicomachean ethics*. [translated by F.H. Peters]. New York: Barnes & Nobles.
- Arrington, M. (2008). Interview with Evan Williams: Summize acquisition, API issues and their revenue model. *TechCrunch*. Retrieved February 6, 2012, from <http://techcrunch.com/2008/07/15/interview-with-evan-william-summize-acquisition-api-issues-and-their-revenue-model>
- Arthur, C. (2011). Twitter angers third-party developers with 'no more timelines' urging. *The Guardian Technology Blog*. Retrieved February 17, 2012, from <http://www.guardian.co.uk/technology/blog/2011/mar/14/twitter-developers-client-warning>
- Ash, J. (2012). Technology, technicity, and emerging practices of temporal sensitivity in videogames. *Environment and Planning A* 44, 187-203.
- Ashby, W. R. (1956). *An introduction to cybernetics*. London: Chapman & Hall.
- Ashton, D. (2011). Upgrading the self: Technology and the self in the digital games perpetual innovation economy. *Convergence*, 17 (3), 307-321.
- Austin, J. L. (1962). *How to do things with words: The William James lectures delivered at Harvard University in 1955*. Cambridge, Mass.: Harvard University Press.
- Bakardjieva, M. (2009). Subactivism: Lifeworld and politics in the age of the internet. *Information Society*, 25 (2), 91-104.
- Baldwin, C. Y., & Clark, K. B. (2000). *Design rules: The power of modularity*. Cambridge, Mass.: MIT Press.
- Banks, J., & Deuze, M. (2009). Co-creative labour. *International Journal of Cultural Studies*, 12(5), 419-431.
- Barad, K. (2007). *Meeting the universe halfway: Quantum physics and the entanglement of matter and meaning*. Durham: Duke University Press.
- Baym, N. K. (2010). *Personal connections in the digital age*. Cambridge: Polity Press.

- Beer, D. (2009). Power through the algorithm? Participatory web cultures and the technological unconscious. *New Media & Society*, 11 (6), 985-1002.
- Benjamin, W., & Arendt, H. (1999). *Illuminations*. London: Pimlico.
- Bernays, E. L. (1947). The Engineering of consent. *Annals of the American Academy of Political and Social Science*, 250: 113-120.
- Berry, D. M. (2008). *Copy, rip, burn: The politics of copyleft and open source*. London: Pluto Press.
- Berry, D. M. (2009). A contribution towards a grammar of code. *Fibreculture*, 13.
- Berry, D. M. (2011). *The philosophy of software: Code and mediation in the digital age*. Houndmills, Basingstoke, Hampshire ; New York: Palgrave Macmillan.
- Blue, V. (2010). Cracking the Facebook news feed code. *ReadWriteWeb*. Retrieved November 28, 2011, from http://www.readwriteweb.com/archives/cracking_the_facebook_news_feed_code.php
- Bodle, R. (2011). Regimes of sharing. open APIs, interoperability, and Facebook. *Information Communication & Society*, 14 (3), 320-337.
- Bogost, I. (2007). *Persuasive games: The expressive power of videogames*. Cambridge, Mass.: MIT Press.
- Boltanski, L., & Chiapello, È. (2005). *The new spirit of capitalism*. London: Verso.
- Bosworth, A. (2007). News feed is a robot! *Facebook blog*. Retrieved February 25, 2012, from <http://blog.facebook.com/blog.php?post=2242467130>
- boyd, d. (2006). Friends, friendsters, and top 8: Writing community into being on social network sites. *First Monday*, 11 (12). Retrieved October 5, 2011, from <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1418/1336>
- boyd, d. (2007). Why youth (heart) social network sites: The role of networked publics in teenage social life. In D. Buckingham (Ed.), *MacArthur Foundation series on digital learning – Youth, identity, and digital media volume*. Cambridge, MA: MIT Press.
- boyd, d. (2008). Facebook's privacy trainwreck: Exposure, invasion, and social convergence. *Convergence*, 14 (1), 13-20.
- boyd, d. (2010). Friendship. In M. Ito, S. Baumer, M. Bittanti, d. boyd, R. Cody, B. Herr, H. Horst, P. Lange, D. Mahendran, K. Martinez, C. J. Pascoe, D. Perkel, L. Robinson, C. Sims & L. Tripp (Eds.), *Hanging out, messing around, and geeking out: Kids living and learning with new media* Cambridge, Mass.: MIT Press.
- boyd, D. M., & Ellison, N. B. (2007). Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13 (1). Retrieved October 5, 2011, from <http://jcmc.indiana.edu/vol13/issue1/boyd.ellison.html>
- Bröckling, U., Krasmann, S., & Lemke, T. (2011). *Governmentality: Current issues and future challenges*. New York: Routledge.
- Brown, S. D. (2004). Parasite logic. *Journal of Organizational Change Management*, 17 (4), 383-395.
- Bruhn Jensen, K. (2012). *A handbook of media and communication research: Qualitative and quantitative methodologies*. London: Routledge.
- Bruns, A. (2008). *Blogs, wikipedia, second life, and beyond: From production to produsage*.

- New York: Peter Lang.
- Buote, V., Wood, E., & Pratt, M. (2009). Exploring similarities and differences between online and offline friendships: The role of attachment style. *Computers in Human Behavior* 25 (2), 560–567.
- Burns, E. (2010). Developing email interview practices in qualitative research. *Sociological Research Online*, 15(4), 8.
- Butler, J. (1990). *Gender trouble: feminism and the subversion of identity*. New York: Routledge.
- Callon, M. (1986). Some elements of a sociology of translation: Domestication of the scallops and the fishermen of St.Brieuc Bay *Sociological Review Monograph*, 196-233.
- Callon, M. (Ed.). (1998). *The laws of the markets*. Oxford: Blackwell.
- Callon, M. (2007). What does it mean to say that economics is performative? In D. MacKenzie, F. Muniesa & L. Siu (Eds.), *Do economics make markets? On the performativity of economics*. Princeton, NJ: Princeton University Press.
- Campbell-Kelly, M. (2003). *From airline reservations to sonic the hedgehog : A history of the software industry*. Cambridge, Mass.: MIT Press.
- Carr, N. G. (2010). *The shallows: What the Internet is doing to our brains*. New York: Norton.
- Ceruzzi, P. E. (2003). *A history of modern computing*. Cambridge, Mass.: MIT Press.
- Chabert, J.L. (1999). *A History of algorithms: From the pebble to the microchip*. New York: Springer.
- Charmaz, K. (2006). *Constructing grounded theory: A practical guide through qualitative analysis*. London: Sage.
- Chen, J., Geyer, W., Dugan, C., Muller, M., & Guy, I. (2009). *Make new friends, but keep the old – Recommending people on social networking sites*. Paper presented at the CHI 09 27th international conference on Human factors in computing systems, New York.
- Chun, W. H. K. (2006). *Control and freedom: Power and paranoia in the age of fiber optics*. Cambridge, Mass.: MIT Press.
- Chun, W. H. K. (2011a). *Programmed visions: Software and memory*. Cambridge, Mass.: MIT Press.
- Chun, W. H. K. (2011b). Crisis, crisis, crisis, or sovereignty and networks. *Theory Culture & Society*, 28 (6), 91-112.
- Clarke, B., & Hansen, M. B. N. (2009). *Emergence and embodiment: New essays on second-order systems theory*: Duke University Press Books.
- Clough, P. T. (2000). *Autoaffection: Unconscious thought in the age of teletechnology*. Minneapolis, Minn.: University of Minnesota Press.
- Cocking, D., & Matthews, S. (2001). Unreal friends. *Ethics and Information Technology* 2 (4), 223-231.
- Constine, J. (2011). Facebook combines most recent and top news into a single feed, adds a real-time news ticker. *Inside Facebook*. Retrieved February 25, 2012, from <http://www.insidefacebook.com/2011/09/20/single-feed-ticker>

- Coonfield, G. (2006). Thinking machinically, or, the techno-aesthetic of Jackie Chan: Toward a Deleuze-Guattarian media studies. *Critical Studies in Media Communication*, 23(4), 285-301.
- Cramer, F. (2005). *Words made flesh: Code, culture, imagination*. Rotterdam: Media Design Research, Piet Zwart Institute.
- Cramer, F. (2008). Language. In M. Fuller (Ed.), *Software studies: A lexicon*. Cambridge, Mass.: MIT Press.
- Cramer, F., & Fuller, M. (2008). Interface. In M. Fuller (Ed.), *Software studies: A lexicon*. Cambridge, Mass.: MIT Press.
- Croghan, P. (2012). *Gameplay mode: War, simulation, and technoculture*: University of Minnesota Press.
- Croghan, P., & Kennedy, H. (2009). Technologies between games and culture. *Games and Culture* 4 (2), 107-114.
- Crutzen, C., & Kotkamp, E. (2008). Object orientation. In M. Fuller (Ed.), *Software studies: A lexicon*. Cambridge, Mass: MIT Press.
- Dahl, O. J., Dijkstra, E. W., & Hoare, C. A. R. (1972). *Structured programming*. London: Academic Press.
- Dahlberg, L. (2007). Rethinking the fragmentation of the cyberpublic: From consensus to contestation. *New Media & Society*, 9 (5), 827-847.
- Daly, E., Geyer, W., & Millen, D. (2010). *The network effects of recommending social connections*. Paper presented at the RecSys'10 The fourth ACM conference on Recommender systems.
- Dayan, D. (2009). Sharing and showing: Television as monstration. *Annals of the American Academy of Political and Social Science*, 625, 19-31.
- de Souza, C., & Bentolila, D. (2009). *Automatic evaluation of API usability using complexity metrics and visualizations*. Paper presented at the 31st International Conference on Software Engineering.
- de Souza, C. R. B., & Redmiles, D. F. (2009). On the roles of APIs in the coordination of collaborative software development. *Computer Supported Cooperative Work-the Journal of Collaborative Computing*, 18 (5-6), 445-475.
- Deleuze, G. (1992). Postscript on the societies of control. *October* (59), 3-7.
- Deleuze, G. (1993). *The fold: Leibniz and the baroque*. London: The Athlone Press.
- Deleuze, G. (2006). *Foucault*. London: Continuum.
- Deleuze, G., & Guattari, F. (1987). *A thousand plateaus: Capitalism and schizophrenia*. Minneapolis, Minn.: University of Minnesota Press.
- Deleuze, G., & Parnet, C. (2007). *Dialogues II*. New York: Columbia University Press.
- Derrida, J. (2005). *The politics of friendship*. London: Verso.
- Dodge, M., & Kitchin, R. (2004). Flying through code/space: The real virtuality of air travel. *Environment and Planning A*, 36 (2), 195-211.
- Dodge, M., & Kitchin, R. (2005). Codes of life: Identification codes and the machine-readable world. *Environment and Planning D-Society & Space*, 23 (6), 851-881.

- Dodge, M., Kitchin, R., & Zook, M. (2009). How does software make space? Exploring some geographical dimensions of pervasive computing and software studies. *Environment and Planning A*, 41 (6), 1283-1293.
- DuVander, A. (2010). Twitter reveals: 75 % of pure traffic is via API (3 billion calls per day). *ProgrammableWeb*. Retrieved February 6, 2012, from <http://blog.programmableweb.com/2010/04/15/twitter-reveals-75-of-our-traffic-is-via-api-3-billion-calls-per-day>
- DuVander, A. (2012). 5000 APIs: Facebook, Google and Twitter are changing the Web. *ProgrammableWeb*. Retrieved February 6, 2012, from <http://blog.programmableweb.com/2012/02/06/5000-apis-facebook-google-and-twitter-are-changing-the-web/>
- Edwards, P. (2002). Infrastructure and modernity: Force, time, and social organization in the history of technical systems. In T. Misa, P. Brey & A. Feenberg (Eds.), *Modernity and Technology*. Cambridge, MA: MIT Press.
- Eilam, E. (2005). *Reversing: Secrets of reverse engineering*. Indianapolis, Ind.: Wiley.
- Ellis, B., Stylos, J., & Myers, B. (2007). *The factory pattern in API design: A usability evaluation*. Paper presented at the ICSE' 07 The 29th International Conference on Software Engineering.
- Ellison, N., Steinfield, C., & Lampe, C. (2007). The benefits of Facebook friends: Social capital and college students' use of online social network sites. *Journal of Computer-Mediated Communication*, 12 (4), 1143-1168.
- Elmer, G. (2003). A diagram of panoptic surveillance. *New Media & Society*, 5 (2), 231-247.
- Elmer, G. (2004). *Profiling machines: Mapping the personal information economy*. Cambridge, Mass.: MIT Press.
- Entman, R. (1993). Framing - Toward clarification of a fractured paradigm. *Journal of Communication*, 43 (4), 51-58.
- Esposito, R. (2010). *Communitas: The origin and destiny of community*. Stanford: Stanford University Press.
- Facebook. (2008). Facebook press release, July 23. Retrieved February 25, 2012, from <http://www.facebook.com/press/releases.php?p=48242>
- Facebook. (2010). Hacking the graph tech talk. *Facebook Engineering*. Retrieved December 19, 2011, from <http://www.facebook.com/video/video.php?v=690842469235>
- Facebook. (2011a). Extending the graph tech talk. *Facebook Engineering*. Retrieved February 10, 2012, from <http://www.facebook.com/video/video.php?v=10150231980165469>
- Facebook. (2011b). How news feed works. Retrieved September 15, 2011, from <http://www.facebook.com/help/?page=408>
- Facebook. (2011c). About news feed. Retrieved November 21, 2011, from <http://www.facebook.com/help/newsfeed>
- Facebook. (2011d). Statistics. Retrieved November 21, 2011, from <http://www.facebook.com/press/info.php?statistics>
- Faubion, J. D. (2001). *Power*. London: Allen Lane.

- Fink, B. (1995). *The Lacanian subject: Between language and jouissance*. Princeton, N.J.: Princeton University Press.
- Foucault, M. (1972). *The archaeology of knowledge and the discourse on language*. New York: Pantheon Books.
- Foucault, M. (1977). *Discipline and punish: The birth of the prison*. London: Allen Lane.
- Foucault, M. (1982). The subject and power. *Critical Inquiry*, 8 (4), 777-795.
- Foucault, M. (1991) 'Governmentality', in G. Burchell, C. Gordon & P. Miller (eds), *The Foucault Effect: Studies in Governmentality*. Hemel Hempstead: Harvester Wheatsheaf.
- Foucault, M., & Rabinow, P. (1997). *The essential works of Michel Foucault, 1954-1984*. London: Allen Lane.
- Foucault, M., Senellart, M., Ewald, F., & Fontana, A. (2007). *Security, territory, population: Lectures at the Collège de France, 1977-78*. Basingstoke: Palgrave Macmillan.
- Foucault, M., Senellart, M., Ewald, F., & Fontana, A. (2008). *The birth of biopolitics: Lectures at the Collège de France, 1978-1979*. Basingstoke: Palgrave Macmillan.
- Franck, G. (1998). *Ökonomie der Aufmerksamkeit: Ein Entwurf*. München: Carl Hanser.
- Fraser, M. (2006). Event. *Theory Culture & Society*, 23 (2-3), 129-132.
- Fuller, M. (2003). *Behind the blip: Essays on the culture of software*. New York: Autonomedia.
- Fuller, M. (2008). *Software studies: A lexicon*. Cambridge, Mass.: MIT Press.
- Galison, P. (1994). The ontology of the enemy: Norbert Wiener and the cybernetic vision *Critical Inquiry*, 21 (1), 228-266.
- Galloway, A. R. (2004). *Protocol: How control exists after decentralization*. Cambridge, Mass.: MIT Press.
- Galloway, A. R. (2006). Language wants to be overlooked: On software and ideology. *Journal of Visual Culture*, 5 (3), 315-331.
- Galloway, A. (2011). Are some things unrepresentable? *Theory Culture & Society*, 28(7-8), 85-102.
- Gane, N. (2005). Radical post-humanism - Friedrich Kittler and the primacy of technology. *Theory Culture & Society*, 22 (3), 25-41.
- Geminder, K. (2007). Platform is here. *Facebook blog*. Retrieved February 25, 2012, from <http://blog.facebook.com/blog.php?post=2437282130>
- Gerlitz, C., & Helmond, A. (2011). Hit, link, like and share. Organizing the social and the fabric of the web in a Like economy, *DMI mini-conference*. Amsterdam.
- Gibson, J. J. (1986). *The ecological approach to visual perception*. Hillsdale, N.J.: Lawrence Erlbaum.
- Gill, R., & Pratt, A. (2008). Precarity and cultural work in the social factory? Immaterial labour, precariousness and cultural work. *Theory Culture & Society*, 25 (7-8), 1-30.
- Gillespie, T. (2007). *Wired shut: Copyright and the shape of digital culture*. Cambridge, Mass.: MIT Press.
- Gillespie, T. (2011). Can an algorithm be wrong? Twitter Trends, the specter of censorship,

- and our faith in the algorithms around us. *Culture Digitally*. Retrieved October 25, 2011, from <http://culturedigitally.org/2011/10/can-an-algorithm-be-wrong/>
- Glaser, B. G., & Strauss, A. L. (1967). *The discovery of grounded theory: strategies for qualitative research*. Chicago: Aldine.
- Goffey, A. (2008). Algorithm. In M. Fuller (Ed.), *Software studies: A lexicon*. Cambridge, Mass.: MIT Press.
- Goffman, E. (1974). *Frame analysis: An essay on the organization of experience*. Cambridge, Mass.: Harvard University Press.
- Goldhaber, M. (1997). The attention economy and the net. *First Monday* 2(4).
- Google. (2011a). Ten algorithm changes on inside search. *Google Blog*. Retrieved March 13, 2012, from <http://googleblog.blogspot.com/2011/11/ten-algorithm-changes-on-inside-search.html>
- Google. (2011b). Finding more high-quality sites in search. *Google Blog*. Retrieved March 13, 2012, from <http://googleblog.blogspot.com/2011/02/finding-more-high-quality-sites-in.html>
- Google. (2011c). Giving you fresher, more recent search results. *Google Blog*. Retrieved March 13, 2012, from <http://googleblog.blogspot.com/2011/11/giving-you-fresher-more-recent-search.html>
- Google. (2012). Technology overview. *Google Company*. Retrieved March 13, 2012, from <http://www.google.com/about/company/tech.html>
- Graham, S. D. N. (2005). Software-sorted geographies. *Progress in Human Geography*, 29(5), 562-580.
- Grier, D. A. (1996). The ENIAC, the verb "to program" and the emergence of digital computers. *Ieee Annals of the History of Computing*, 18 (1), 51-55.
- Grimmelmann, J. (2009). Saving Facebook. *Iowa Law Review*, 94 (4), 1137-1206.
- Grossberg, L. (1986). On postmodernism and articulation : An interview with Stuart Hall. *Journal of Communication Inquiry*, 10, 45-60.
- Guattari, F. (1989). The three ecologies. *New Formations*, 8, 131-147.
- Guattari, F. (1996). Subjectivities: For better and for worse. In G. Genosko (Ed.), *The Guattari reader*. Oxford: Blackwell.
- Guillory, J. (2010). Genesis of the media concept. *Critical Inquiry*, 36(2), 321-362.
- Gumbrecht, H. U. (2004). *Production of presence: what meaning cannot convey*. Stanford, Calif.: Stanford University Press.
- Hacking, I. (1995). The looping effects of human kinds. In D. Sperber, D. Premack & A. Premack (Eds.), *Causal cognition: A multidisciplinary approach*. Oxford: Clarendon Press.
- Hampton, K. N., Goulet, S. L., Rainie, L., & Purcell, K. (2011). Social networking sites and our lives. *Pew Internet & American Life Project*. Retrieved January 11, 2012, from <http://pewinternet.org/Reports/2011/Technology-and-social-networks.aspx/>
- Halonen, R. (2007). Users As Developers In Information System Projects. *Observatorio (OBS*) Journal*, 3: 115-130 1646-

- Hansen, M. B. N. (2012). Ubiquitous sensibility. In J. Packer & S. Wiley Crofts (Eds.), *Communication matters: Materialist approaches to media, mobility and networks*. London: Routledge.
- Hardt, M., & Negri, A. (2000). *Empire*. Cambridge, Mass.: Harvard University Press.
- Hargittai, E. (2007). The social, political, economic, and cultural dimensions of search engines: An introduction. *Journal of Computer-Mediated Communication*, 12 (3).
- Harman, G. (2009). *Prince of networks: Bruno Latour and metaphysics*. Melbourne re.press.
- Hayles, N. K. (1999). *How we became posthuman: Virtual bodies in cybernetics, literature, and informatics*. Chicago: University of Chicago Press.
- Hayles, N. K. (2004). Print is flat, code is deep: The importance of media-specific analysis. *Poetics Today*, 25 (1), 67-90.
- Hayles, N. K. (2005). *My mother was a computer: Digital subjects and literary texts*. Chicago: University of Chicago Press.
- Hayles, N. K. (2007). Hyper and deep attention: The generational divide in cognitive modes. *Profession*, 13, 187–199.
- Hays, R. (1988). Friendship. In S. Duck (Ed.), *Handbook of personal relationships: Theory, Research, and Interventions*. New York: Wiley.
- Heidegger, M. (1977). *The question concerning technology and other essays*. New York: Harper & Row.
- Hellsten, I., Leydesdorff, L., & Wouters, P. (2006). Multiple presents: How search engines rewrite the past. *New Media & Society*, 8 (6), 901-924.
- Higginbotham, S. (2011). Are APIs the new black? *Gigaom*. Retrieved February 21, 2012, from <http://gigaom.com/2011/03/16/are-apis-the-new-black/>
- Hine, C. (2000). *Virtual ethnography*. London: Sage.
- Hine, C. (2011). Towards ethnography of television on the internet: A mobile strategy for exploring mundane interpretive activities. *Media Culture & Society*, 33 (4), 567-582.
- Hoffman, M. (2011). Disciplinary power. In D. Taylor (Ed.), *Michel Foucault key concepts*. Durham: Acumen.
- Hull, M. (2011). Facebook changes mean that you are not seeing everything that you should be seeing. Retrieved April 20, 2011, from <http://www.facebook.com/notes/mark-hull/please-read-facebook-changes-mean-that-you-are-not-seeing-everything-that-you-sh/10150089908123789>
- Introna, L. D., & Nissenbaum, H. (2000). Shaping the Web: Why the politics of search engines matters. *Information Society*, 16 (3), 169-185.
- Jansen, K., & Vellema, S. (2011). What is technography? *Njas-Wageningen Journal of Life Sciences*, 57 (3-4), 169-177.
- Jenkins, H. (2006). *Convergence culture: Where old and new media collide*. New York: New York University Press.
- Joinson, A. N. (2008). 'Looking at', 'looking up' or 'keeping up with' people? *Motives and uses of Facebook*. Paper presented at the CHI'08 26th Annual Conference on Human Factors in Computing Systems.

- Kelty, C. M. (2008). *Two bits: The cultural significance of free software*. Durham: Duke University Press.
- Kien, G. (2008). Technography = Technology plus ethnography an introduction. *Qualitative Inquiry*, 14 (7): 1101-1109.
- Kincaid, J. (2010a). EdgeRank: The secret sauce that makes Facebook's news feed tick. *TechCrunch*. Retrieved February 3, 2012, from <http://techcrunch.com/2010/04/22/facebook-edgerank>
- Kincaid, J. (2010b). Twitter acquires Tweetie. *TechCrunch*. Retrieved February 17, 2012, from <http://techcrunch.com/2010/04/09/twitter-acquires-tweetie>
- Kirkpatrick, M. (2008). APIs and developer platforms: A discussion on the pros and cons. *ReadWriteWeb*. Retrieved February 28, 2012, from http://www.readwriteweb.com/archives/apis_platforms_pros_and_cons.php
- Kirschenbaum, M. G. (2008). *Mechanisms: New media and the forensic imagination*. Cambridge, Mass.: MIT Press.
- Kitchin, R., & Dodge, M. (2011). *Code/space: software and everyday life*. Cambridge, Mass.: MIT Press.
- Kittler, F. A. (1990). *Discourse networks 1800/1900*. Stanford, Calif.: Stanford University Press.
- Kittler, F. A. (1999). *Gramophone, film, typewriter*. Stanford, Calif.: Stanford University Press.
- Kittler, F. A., & Johnston, J. (1997). *Literature, media, information systems: essays*. Amsterdam: G+B Arts International.
- Knuth, D. E. (1968). *The art of computer programming*. Reading, Mass.: Addison-Wesley.
- Kracauer, S., & Levin, T. Y. (1995). *The mass ornament: Weimar essays*. Cambridge, Mass.: Harvard University Press.
- Krämer, S. (2006). The Cultural techniques of time axis manipulation. On Friedrich Kittler's conception of media. *Theory Culture & Society*, 23 (7-8), 93-109.
- Kvale, S. (1996). *Interviews: An introduction to qualitative research interviewing*. Thousand Oaks, Calif.: Sage.
- Langlois, G. (2012). Participatory Culture and the New Governance of Communication. *Television & New Media*. Online first February 2.
- Langlois, G., & Elmer, G. (2009). Wikipedia leeches? The promotion of traffic through a collaborative web format. *New Media & Society*, 11 (5), 773-794.
- Langlois, G., Elmer, G., McKelvey, F., & Devereaux, Z. (2009a). Networked publics: The double articulation of code and politics on Facebook. *Canadian Journal of Communication*, 34, 415-434.
- Langlois, G., McKelvey, F., Elmer, G., & Werbin, K. (2009b). Mapping commercial Web 2.0 worlds: Towards a new critical ontogenesis. *Fibreculture* (14).
- Latour, B. (1988). *The pasteurization of France*. Cambridge, Mass.: Harvard University Press.
- Latour, B. (1994). On technical mediation – philosophy, sociology, genealogy. *Common Knowledge*, 3 (2), 29-64.

- Latour, B. (2005). *Reassembling the social: An introduction to actor-network-theory*. Oxford: Oxford University Press.
- Law, J. (1992). The olympus 320 engine: A case-study in design, development, and organizational control. *Technology and Culture*, 33 (3), 409-440.
- Law, J. (2002). Objects and spaces. *Theory Culture & Society*, 19 (5-6), 91-105.
- Lazzarato, M. (2004). From capital-labour to capital-life. *Ephemera*, 4 (3), 187-208.
- Lazzarato, M. (2006). The concepts of life and the living in the societies of control. In M. Fuglesang & B. Sørensen (Eds.), *Deleuze and the social*. Edinburgh: Edinburgh University Press.
- Lazzarato, M. (2007). From the revolutions of capitalism. *SubStance*, 36 (1).
- Leigh-Star, S. (1999). The ethnography of infrastructure. *American Behavioral Scientist* 43(3), 377-391.
- Leistert, O., & Röhle, T. (2011). *Generation Facebook: Über das Leben im Social Net*. Bielefeld: Transcript Verlag.
- Lemke, T. (2001). The birth of bio-politics – Michel Foucault’s lecture at the Collège de France on neo-liberal governmentality. *Economy & Society*, 30 (2), 190-207.
- Lennon, A. (2009). A conversation with Twitter co-founder Jack Dorsey. *The Daily Anchor*. Retrieved February 6, 2012, from <http://www.thedailyanchor.com/2009/02/12/a-conversation-with-twitter-co-founder-jack-dorsey>
- Lessig, L. (1999). *Code: And other laws of cyberspace*. New York: Basic Books.
- Levy, S. (2010). *Hackers*. Sebastopol, Calif.: O'Reilly Media.
- Lewin, K. (1947). Frontiers in group dynamics II - Channels of group life; Social planning and action research. *Human Relations*, 1 (2), 143-153.
- Lewis, J., & West, A. (2009). ‘Friending’: London-based undergraduates’ experience of Facebook. *New Media & Society*, 11 (7), 1209-1229.
- Lingis, A. (1994). *The community of those who have nothing in common* Bloomington, Ind: Indiana University Press.
- Lovink, G. (2007). *Zero comments: Blogging and critical Internet culture*. New York: Routledge.
- Løvlie, A. (2011). *Textopia: Experiments with locative literature*. University of Oslo, Oslo. Unpublished Ph.D. dissertation.
- Lüders, M. (2009). Becoming more like friends: A qualitative study of personal media and social life. *Nordicom Review*, 30 (1), 201-216.
- Lynch, S. (2005). *Philosophy and friendship*. Edinburgh: Edinburgh University Press.
- Mackenzie, A. (2002). *Transductions: Bodies and machines at speed*. London: Continuum.
- Mackenzie, A. (2005). The performativity of code - Software and cultures of circulation. *Theory Culture & Society*, 22 (1), 71-92.
- Mackenzie, A. (2006). *Cutting code: Software and sociality*. New York: Peter Lang.
- Mackenzie, A. (2007). Protocols and the irreducible traces of embodiment: The viterbi algorithm and the mosaic of machine time. In R. Hassan & R. Purser (Eds.), *24/7: time and temporality in the network society*. Stanford: Stanford University Press.

- Mackenzie, A., & Vurdubakis, T. (2011). Codes and codings in crisis signification, performativity and excess. *Theory Culture & Society*, 28 (6), 3-23.
- Madrigal, A. (2010). How the Facebook news feed algorithm shapes your friendships. *The Atlantic*. Retrieved November 18, 2011, from <http://www.theatlantic.com/technology/archive/2010/10/how-the-facebook-news-feed-algorithm-shapes-your-friendships/64996/>
- Manovich, L. (2001). *The Language of New Media*. Cambridge, Mass.: MIT Press.
- Manovich, L. (2008). *Software Takes Command*. Retrieved from May 2, 2011, from <http://lab.softwarestudies.com/2008/11/softbook.html>
- Manovich, L. (2011). Cultural Software. Retrieved March 15, 2012, from <http://manovich.net/2011/07/14/new-article-cultural-software-lev-manoich-7142011>
- Manovich, L. (2012). How to follow software users? Retrieved April 5, 2012, from <http://lab.softwarestudies.com/p/publications.html>
- Marcus, G. E. (1995). Ethnography in/of the world-system: The emergence of multi-sited ethnography *Annual Review of Anthropology*, 24, 95-117.
- Marino, M. (2006). Critical code studies. Retrieved February 15, 2012, from <http://www.electronicbookreview.com/thread/electropoetics/codology/>
- Marwick, A. (2010). Status update: Celebrity, publicity and self-branding in Web 2.0. Unpublished PhD dissertation.
- Massumi, B. (2002). *Parables for the virtual: Movement, affect, sensation*. Durham, N.C.: Duke University Press.
- Massumi, B. (2009). 'Technical materiality' revisited: Brian Massumi on Gilbert Simondon. *Parrhesia*, 7, 36-45.
- Mateas, M., & Wardrip-Fruin, N. (2009). *Defining operational logics*. Paper presented at the Digital Games Research Association (DiGRA).
- McCombs, M. E., & Shaw, D. L. (1972). Agenda-setting function of mass media. *Public Opinion Quarterly*, 36(2), 176-&.
- McKelvey, F. (2010). Ends and ways: The algorithmic politics of network neutrality. *Global Media Journal — Canadian Edition*, 3 (1), 51-73.
- McKelvey, F. (2011). A Programmable Platform? Drupal, Modularity, and the Future of the Web. *Fibreculture*, 18.
- McLuhan, M. (1994). *Understanding media: The extensions of man*. Cambridge, Mass.: MIT Press.
- McLuhan, M., & Fiore, Q. (1967). *The medium is the massage: An inventory of effects*. New York: Bantam books.
- Metz, C. (2012). API: Three letters that change life, the universe and even Detroit. *Wired*. Retrieved February 16, 2012, from <http://www.wired.com/wiredenterprise/2012/02/apis-change-the-world/all/1>
- Meyrowitz, J. (1998). Multiple media literacies. *Journal of Communication*, 48 (1), 96-108.
- Michael, M. (2004). On making data social: Heterogeneity in sociological practice. *Qualitative Research*, 4(1), 5-23.

- Miller, P., & Rose, N. (1990). Governing economic life. *Economy and Society*, 19 (1), 1-31.
- Miller, P., & Rose, N. (2008). *Governing the present: Administering economic, social and personal life*. Cambridge: Polity.
- Mitchell, W. J. T. & Hansen, M. B. N. (2010). *Critical terms for media studies*. Chicago: University of Chicago Press.
- Mol, A., & Law, J. (1994). Regions, networks and fluids - Anemia and social topology. *Social Studies of Science*, 24 (4), 641-671.
- Moricz, M., Dosbayev, Y. & Berlyant, M. (2010). *PYMK: Friend recommendation at MySpace*. Paper presented at the SIGMOD '10. The 2010 international conference on Management of data, Indianapolis, Indiana, USA.
- Munster, A. (2011). Nerves of data: The neurological turn in/against networked media. *Computational culture: a journal of software studies*, 1. Retrieved February 16, 2012, from <http://computationalculture.net/article/nerves-of-data>
- Musser, J. (2007). Twitter API traffic is 10x Twitter's site. *ProgrammableWeb*. Retrieved February 6, 2012, from <http://blog.programmableweb.com/2007/09/10/twitter-api-traffic-is-10x-twitthers-site>
- Nakamura, L. (2002). *Cybertypes*. New York: Routledge.
- Naughton, J. (2012). Why all our kids should be taught how to code. *The Guardian*. Retrieved April 10, 2012, from <http://www.guardian.co.uk/education/2012/mar/31/why-kids-should-be-taught-code>
- Neff, G., Wissinger, E., & Zukin, S. (2005). Entrepreneurial labor among cultural producers: "Cool" jobs in "hot" industries. *Social Semiotics*, 15 (3), 307-334.
- Newman, M. (2010). New media, young audiences and discourses of attention: From Sesame Street to 'snack culture'. *Media, Culture & Society*, 32(4), 581-596.
- Niederer, S. & van Dijck, J. (2010). Wisdom of the crowd or technicity of content? Wikipedia as a sociotechnical system. *New Media & Society*, 12 (8), 1368-1387.
- O'Reilly, T. (2005). What is Web 2.0. *O'Reilly*. Retrieved 8 February, 2012, from <http://oreilly.com/pub/a/web2/archive/what-is-web-20.html?page=1>
- Ong, W. J. (2002). *Orality and literacy: The technologizing of the word*. London: Routledge.
- Packer, J., & Wiley, S. B. C. (2012). *Communication matters: Materialist approaches to media, mobility, and networks*. London: Routledge.
- Parikka, J. (2011a). New materialism as media theory: Medianatures and dirty matter. *Communication and Critical/Cultural Studies* 1-6.
- Parikka, J. (2011b). Operative media archaeology: Wolfgang Ernst's materialist media diagrammatics. *Theory Culture & Society*, 28 (5), 52-74.
- Parks, M., & Roberts, L. (1998). Making moosic: The development of personal relationships on-line and a comparison to their off-line counterparts. *Journal of Social and Personal Relationships*, 15 (4), 517-537.
- Parr, B. (2009). The evolution of the social media API. *Mashable*. Retrieved February 28, 2012, from <http://mashable.com/2009/05/21/social-media-api>
- Parr, B. (2011a). Twitter acquires TweetDeck. *Mashable*. Retrieved February 17, 2012, from <http://mashable.com/2011/05/25/twitter-acquires-tweetdeck>

- Parr, B. (2011b). Facebook turns friend activity into new ad format. *Mashable*. Retrieved November 21, 2011, from <http://mashable.com/2011/01/25/facebook-sponsored-stories>
- Pashler, H. E. (1998). *The psychology of attention*. Cambridge, Mass.: MIT Press.
- Pasquinelli, M. (2009). Google's PageRank algorithm: A diagram of the cognitive capitalism and the rentier of the common intellect. In K. Becker & F. Stalder (Eds.), *Deep Search*. London: Transaction Publishers.
- Pels, D., Hetherington, K., & Vandenberghe, F. (2002). The status of the object - Performances, mediations, and techniques. *Theory Culture & Society*, 19 (5-6), 1-21.
- Perez, S. (2010). Twitter is not a social network, says Twitter exec. *ReadWriteWeb*. Retrieved February 7, 2012, from http://www.readwriteweb.com/archives/twitter_is_not_a_social_network_says_twitter_exec.php
- Peters, J. D., & Packer, J. (2012). Becoming mollusk: A conversation with John Durham Peters about media, materiality, and matters of history. In J. Packer & S. Wiley Crofts (Eds.), *Communication matters: Materialist approaches to media, mobility and networks*. London: Routledge.
- Phillips, J. (2006). Agencement/Assemblage. *Theory, Culture & Society*, 23, 108-109.
- Pickering, A. (1995). *The mangle of practice: time, agency, and science*. Chicago: University of Chicago Press.
- Quintana, M. (2010). Facebook tips: What's the difference between top news and most recent? *Facebook Blog*. Retrieved January 20, 2012, from <http://blog.facebook.com/blog.php?post=414305122130>
- Rabinow, P. (1999). *French DNA: Trouble in purgatory*. Chicago: University of Chicago Press.
- Rajchman, J. (1988). Foucault art of seeing. *October*(44), 89-107.
- Rancière, J. (2004). *The politics of aesthetics: The distribution of the sensible*. London: Continuum.
- Riemer, F. (2009). Ethnography research. In S. Laplan & M. Quartaroli (Eds.), *Research essentials: An introduction to designs and practices*. Jossey-Bass.
- Rodowick, D. N. (1990). Reading the figural. *Camera Obscura* (24), 11-45.
- Rose, N. (1999). *Powers of freedom: Reframing political thought*. Cambridge: Cambridge University Press.
- Ross, C., Orr, E. S., Sisic, M., Arseneault, J. M., Simmering, M. G., & Orr, R. R. (2009). Personality and motivations associated with Facebook use. *Computers in Human Behavior*, 25 (2), 578-586.
- Rubin, H. J., & Rubin, I. (1995). *Qualitative interviewing: The art of hearing data*. Thousand Oaks, Calif.: Sage.
- Schroeder, S. (2009). Twitter API Gets Rate Limit; Will It Hurt App Growth? *Mashable*. Retrieved February 16, 2012, from <http://mashable.com/2009/01/21/twitter-api-gets-limited/>
- Scriptol. (2012). Definition of algorithm. *Scriptol*. Retrieved April 15, 2012, from

- <http://www.scriptol.com/programming/algorithm-definition.php>
- Seel, M. (2000). *Ästhetik des Erscheinens*. München: Hanser.
- Seligstein, J. (2010). See the messages that matter. *Facebook Blog*. Retrieved January 20, 2012, from <http://blog.facebook.com/blog.php?post=452288242130>
- Serres, M. (1982). *The parasite*. Baltimore: The John Hopkins University Press.
- Serres, M. (1995). *Genesis*. Ann Arbor: University of Michigan Press.
- Shahani, S., Roshon, N., Sharma, S., & Leonard, A. (2011). *How to make your brand more visible in the new Facebook*. Marketing report. iCrossing. Retrieved November 25, 2011, from <http://icrossing.com>
- Siegler, M. (2011). Twitter drops the ecosystem hammer: Don't try to compete with us on clients, focus on data and verticals. *TechCrunch*. Retrieved February 26, 2012, from <http://techcrunch.com/2011/03/11/twitter-ecosystem-guidelines>
- Simon, H. (1971). Designing organizations for an information-rich world. In M. Greenberger (Ed.), *Computers, communication, and the public interest*. Baltimore, MD: The Johns Hopkins Press.
- Simondon, G. (1992). The genesis of the individual. In J. Crary & S. Kwinter (Eds.), *Incorporations*. New York: Zone
- Simondon, G. (2009). Technical mentality. *Parrhesia*, 7 (17-27).
- Song, F. (2010). Theorizing web 2.0. *Information, Communication and Society*, 13 (2), 249-275.
- Stiegler, B. (1998). *Technics and time*. Stanford, Calif.: Stanford University Press.
- Stiegler, B. (2010). *Taking care of youth and the generations*. Standford, Calif.: Stanford University Press.
- Sullivan, D. (2011). By the numbers: How Facebook says likes & social plugins help websites. *Search Engine Land*. Retrieved February 25, 2012, from <http://searchengineland.com/by-the-numbers-how-facebook-says-likes-social-plugins-help-websites-76061>
- Taylor, B. (2010). The Next Evolution of Facebook Platform. *Facebook Developers Blog*. Retrieved February 25, 2012, from <http://developers.facebook.com/blog/post/377>
- Taylor, B. (2011). f8 developers conference keynote. Retrieved March 16, 2012, from http://www.facebook.com/f8/app_283743208319386
- Taylor, D. (2011). Everything you need to know about Facebook's EdgeRank. *TheNextWeb*. Retrieved February 3, 2012, from <http://thenextweb.com/socialmedia/2011/05/09/everything-you-need-to-know-about-facebook's-edgerank>
- Telfer, E. (1991). Friendship. In M. Pakaluk (Ed.), *Other selves: Philosophers on friendship*. Indianapolis: Hackett.
- Terranova, T. (2000). Free Labor: Producing culture for the digital economy. *Social Text*, 18(2), 33-58.
- Terranova, T. (2004). *Network culture: Politics for the information age*. London: Pluto Press.
- Thompson, J. B. (2005). The new visibility. *Theory Culture & Society*, 22(6), 31-+.

- Thrift, N. (2004). Remembering the technological unconscious by foregrounding knowledges of position. *Environment and Planning D-Society & Space*, 22 (1), 175-190.
- Thrift, N. (2005). *Knowing capitalism*. London: Sage.
- Thrift, N. (2008). *Non-representational theory: Space, politics, affect*. London: Routledge.
- Thrift, N., & French, S. (2002). The automatic production of space. *Transactions of the Institute of British Geographers*, 27 (3), 309-335.
- Tonkelowitz, M. (2011). Interesting news, any time you visit. *The Facebook Blog*. Retrieved February 3, 2012, from <http://blog.facebook.com/blog.php?post=10150286921207131>
- Trendwatching. (2011). The F-Factor. *Trendwatching*. Retrieved May 25, 2011, from <http://trendwatching.com/trends/ffactor>
- Twitter. (2011). One million registered Twitter apps. *Twitter Blog*. Retrieved February 26, 2012, from <http://blog.twitter.com/2011/07/one-million-registered-twitter-apps.html>
- Uprichard, E., Burrows, R., & Parker, S. (2009). Geodemographic code and the production of space. *Environment and Planning A*, 41 (12), 2823-2835.
- Valenzuela, S., Park, N., & Kee, K. F. (2009). Is there social capital in a social network site?: Facebook use and college students' life satisfaction, trust, and participation. *Journal of Computer-Mediated Communication*, 14 (4), 875-901.
- Vallor, S. (2011). Flourishing on facebook: Virtue friendship & new social media. *Ethics and Information Technology*. Online first January 7.
- van Dijck, J. (2012). Facebook as a tool for producing sociality and connectivity. *Television & New Media*, 13(2), 160-176.
- von Hilgers, P. (2009). Ursprünge der Black Box. In P. von Hilgers & A. Ofak (Eds.), *Rekursionen: Von Faltungen des Wissens* (pp. 127-145). Berlin: Fink.
- Vannini, P., & Vannini, A. (2008). Of walking shoes, boats, golf carts, bicycles, and a slow technoculture: A technography of movement and embodied media on protection island. *Qualitative Inquiry*, 14 (7), 1272-1301.
- Wallimann, I., Tatsis, N., & Zito, G. (1977). On Max Weber's definition of power. *Journal of Sociology*, 13 (3), 231-235.
- Walter, E. (2011). 10 tips for posting on your brand's Facebook Page. *Mashable*. Retrieved February 25, 2012, from <http://mashable.com/2011/03/22/tips-brand-facebook-page>
- Wardrip-Fruin, N. (2006). *Expressive processing: On process-intensive literature and digital media*. Brown University.
- Wardrip-Fruin, N. (2009). *Expressive processing: Digital fictions, computer games, and software studies*. Cambridge, Mass.: MIT Press.
- Webb, D. (2003). On friendship: Derrida, Foucault, and the practice of becoming. *Research in Phenomenology* 33 (1), 119-140.
- Webster, J. (2011). The duality of media: A structural theory of public attention. *Communication Theory* 21 (1), 43-66.
- Whitehead, A. N. (1978). *Process and reality*. Edited by Griffin, D. R., & Sherburne, D. W. New York: Free Press.
- Winner, L. (1986). *The whale and the reactor: A search for limits in an age of high*

- technology*. Chicago: University of Chicago Press.
- Winthrop-Young, G., & Gane, N. (2006). Friedrich Kittler - An introduction. *Theory Culture & Society*, 23 (7-8), 5-+.
- Wise, J. (2012). Attention and assemblage in the clickable world. In J. Packer & S. Wiley Crofts (Eds.), *Communication matters: Materialist approaches to media, mobility and networks*. London: Routledge.
- Wolf, M. (2007). *Proust and the squid: The story and science of the reading brain*. New York: HarperCollins.
- Wortham, J. (2012). A surge in learning the language of the Internet. *New York Times*. Retrieved April 10, 2012, from <http://www.nytimes.com/2012/03/28/technology/for-an-edge-on-the-internet-computer-code-gains-a-following.html?pagewanted=all>
- Zhao, S. Y., Grasmuck, S., & Martin, J. (2008). Identity construction on Facebook: Digital empowerment in anchored relationships. *Computers in Human Behavior*, 24 (5), 1816-1836.
- Zook, M., & Graham, M. (2007). The creative reconstruction of the Internet: Google and the privatization of cyberspace and DigiPlace. *Geoforum*, 38, 1322-1343
- Zuckerberg, M. (2010). Building the social web together. *Facebook Blog*. Retrieved January 20, 2012, from <http://blog.facebook.com/blog.php?post=383404517130>
- Zuckerberg, M. (2011). F8 developers conference keynote. Retrieved March 16, 2012, from http://www.facebook.com/f8/app_283743208319386

Appendix 1. List of interviews

Informant	Date of e-mails received
Jack	10.8.2010; 12.8.2010
Edward	10.8.2010; 12.08.2010
Thomas	11.8.2010
Peter	11.8.2010; 28.7.2011; 20.1.2012
Brad	11.8.2010
Patrick	20.8.2010
Henry	3.9.2010
Tony	21.9.2010; 15.6.2011
Rob	21.9.2010
Alex	25.9.2010; 17.6.2011
Alexandra	26.9.2010
Joey	8.11.2010; 15.11.2010
Eric	15.6.2011; 16.06.2011
Daniel	15.6.2011
Julian	16.6.2011
Carl	20.6.2011
Jacob	1.8.2011; 2.8.2011
Oliver	3.8.2011
Nick	5.8.2011; 12.09.2011
Jeremy	9.8.2011
Matt	19.8.2011

All names have been changed to protect the informants' privacy.