WIZARDS OF THE WEB

AN OUTSIDER'S JOURNEY INTO TECH CULTURE, PROGRAMMING, AND MATHEMAGICS

JAKOB SVENSSON

NORDICOM

WIZARDS OF THE WEB

WIZARDS OF THE WEB

AN OUTSIDER'S JOURNEY INTO TECH CULTURE, PROGRAMMING, AND MATHEMAGICS

Jakob Svensson

NORDICOM

Wizards of the Web

An Outsider's Journey into Tech Culture, Programming, and Mathemagics

Jakob Svensson

ISBN 978-91-88855-52-7 (print)
ISBN 978-91-88855-53-4 (pdf)

DOI: https://doi.org/10.48335/9789188855534

The publication is also available Open Access at www.nordicom.gu.se

Published by:
Nordicom
University of Gothenburg
Box 713
SE 405 30 GÖTEBORG
Sweden

Cover by: Charlotta Hammar

To mum and dad

"Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. And while some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do".

– from Apple's "Think Different" campaign,
written by creative director Rob Siltanen with Ken Segal
(and a bunch of other Apple employees)

Contents

Preface

My name is Jakob Svensson, and I am a media and communication researcher in the south Swedish town of Malmö. In 2018, I received a research grant from the Swedish Research Council to study the people behind data, algorithms, and automated systems. In the grant application, I had written that my project would contribute with a much-needed sociological approach by focusing on the human and cultural aspects of contemporary data societies. Being engineered by programmers, algorithms and automated system are marked by rules, ideals, imaginations, and cultures. They are encoded with human intentions that may or may not be fulfilled. This book is about the humans behind the screen. A beloved child has many names, as the Swedish proverb goes, and I use "programmers" as an umbrella term for coders, web developers, and software engineers. While the programmers are at centre stage, this book is also about my journey into tech culture, travelling to conferences and tech centres around the world.

There are many that must be thanked for this book. First and foremost, Johannes, my editor, who with great interest and detail, and not the least, a good mood, has pushed me to be better and to kill my darlings. Thanks also to Kristin for helping me with my English, and of course the Swedish Research Council (Swedish tax payers) for funding this research. I also want to thank my university departments: Informatics and Media at Uppsala University, where it all started, and the school of Arts and Communication (K3) at Malmö University, where I am currently based. (A special thanks to the latter for making space for me and this project at times of stress, depression, and a burnout.) There are many colleagues at these departments who have provided insightful comments during the years. I have also had the privilege to present this research at the Weizenbaum Institute (Berlin), the American University (Rome), Monash University (Melbourne), and the universities in Umeå, Gävle, and Södertörn (all in Sweden).

Most of all, however, thanks to my friends and family. Without you this wouldn't have been possible. I wrote this book at a very difficult time in my life and your support and care meant the world to me. This book is dedicated to my parents – two crazy ones who really dared to be different and showed me early on how to follow your heart, even if that meant swimming against the tide.

Chapter 1

Introduction
The humans behind the screen

It's mid-March 2019 and I find myself in a spacious AirBNB house in the eastern suburbs of Austin. The sun is shining and the air is warming up after a cool night – at least it's a lot warmer than Malmö, I think to myself. Here to attend the legendary and exciting South by Southwest (SXSW) festival, I still maintain my daily morning habits from back home – that is, to start the day listening to Swedish Radio's morning broadcast of public service channel P1. Two news stories catch my attention. The first one is about the aftermath of the crash of Ethiopian Airlines Flight 302 to Nairobi a couple of days earlier. Apparently, the autopilot didn't work as it should: Boeing's newly developed automatic system that should prevent the aircraft from stalling failed – and the aircraft crashed. This wasn't the first time software bugs led to disaster. In 1962, five minutes after its launch, the Mariner I spacecraft veered off course and had to be blown up to prevent it from crashing in an inhabited area. The problem? A missing hyphen in its guidance control program (Wikipedia, 2021b).

We put our lives in the hands of data-fed, so-called automated systems – systems that we who are not versed in computer programming cannot fully understand, but where power is increasingly accumulated. In late modern risk societies – in which people increasingly depend on each other's specialised skills (see Beck, 1992; Elias, 1939/1998) – this might not be so suprising. What is more intriguing is how little we know about the people behind the systems we depend upon: Who are they? Where do they come from? What norms, values, and imaginations govern their work?

The other story that caught my attention that morning in Austin was about bullying and online expressions of hate, which apparently are increasing among teenagers with the help of communication apps that allow users to leave anonymous comments about each other. The service in this news story's line of fire was Tellonym, an app which appears to be popular among Swedish high-schoolers. According to Tellonym, the service "allows you to receive anonymous and honest feedback from everyone who is important to you" (Tellonym, n.d.). The line between honest feedback and bullying, however, seems to be very thin. Representatives from Tellonym were unwilling to be interviewed by Swedish Radio, but answered in an e-mail that they take responsibility through their

moderation filters and algorithms. I was stunned. Tellonym representatives actually outsource responsibility and ethics to algorithms, washing their hands of any responsibility as if these systems were independent – as if they were separate from the service and the company's business model, unamendable, neutral, and untouched by humans and their preconceptions?

Having made myself familiar with algorithms and automated systems for my research application to the Swedish Research Council, I understood the importance of the humans behind them and the often predefined and specific problems their programming was supposed to solve. In the Tellonym case, it was not to prevent online bullying, but most likely to attract teenagers to the service for commercial purposes – in other words, to harvest data from their users, either to sell to a third party or to use it to target advertisements. Online bullying might be an *unintended* consequence of the app (even though I find it hard to believe that the Tellonym programmers were unable to foresee online bullying as a possible consequence of their service); however, being unintended does not mean online bullying can't be prevented. Apparently, the algorithms, which Tellonym's programmers engineered, were doing a poor job in this regard.

What these news stories exemplify is how automated systems – or algorithms making calculations on predefined sets of data – are becoming increasingly important. Software designer Christopher Steiner was early to point this out in his 2012 book *Automate This,* with the telling subtitle, *How Algorithms Came to Rule the World*. Steiner argues that algorithms control financial markets, what music reaches our ears, and even how we choose our partners. By deciding which data to include in their calculations, algorithms determine our creditworthiness, evaluate our job applications, and estimate the likelihood of us being victims – or perpetrators – of violence. Algorithms are increasingly responsible for selecting the information that reaches people in connected societies, thus setting as well as framing the agenda, which in turn influences decisions and preferences. As Steiner (2012: 6) argues, algorithms are potent tools "we use to shape our planet, our lives and even our culture".

Indeed, the outcomes of algorithmic calculations have consequences not only for signed-in platform users, but for citizens in general. A case in point is the police using algorithms for crime prevention. Media scholar Lina Dencik and her collaborators in the EU-funded Data Justice project have provided examples of how Big Data and social media are used for so-called predictive policing in the UK (see, e.g., Dencik et al., 2016). The case of predictive policing underlines that data uses have shifted from reactive to proactive forms of governance. Mathematician Cathy O'Neill – in her bestselling book *Weapons of Math Destruction* (2016) – shows how predictive policing systems send police officers back to the same poor neighbourhoods, creating a toxic feedback loop because policing one street creates new data that justifies more policing in

that same street. Since every police arrest creates more data, Latinos and Black individuals become more likely to be stopped in crime prevention measures in the US. At the same time, economic (predominantly White) crime often goes unseen. O'Neill thus concludes that predictive policing zeroes in on the poor, resulting in the criminalisation of poverty and the belief that this is scientific and fair. In addition, as political scientist Virginia Eubanks underlines in her award-winning book *Automating Inequality* (2017), while people in connected societies live under this new regime of data and automated systems, the most invasive and punitive systems zoom in on the poor.[1]

Given their increasing power, it's surprising how little we in front of the screen know about these systems and their makers. While our actions are increasingly rendered into data, most of us have little to no idea of just how much data is collected, what it's used for, or who has access to it. This is a one-way mirror with a fundamental lack of transparency, as legal scholar Frank Pasquale argues in his book *The Black Box Society* (2015). Hence, algorithms and automated systems are notoriously difficult to hold accountable. Perhaps this is why the Tellonym representative in the radio news story outsourced responsibility to filters and algorithms, since they give the impression of being untouched by human hands, and thus also by the institutional and cultural circumstances in which they were designed and engineered.

Tellonym is not the first social media company who has attempted to divert attention from themselves by pointing at their algorithms (as if they could be separated from their sociocultural and institutional setting). A well-known example is Google's "image search algorithm": a search for "three white teenagers" resulted in a multitude of (for sale) stock photography, while a search for "three black teenagers" resulted in arrest mugshots. Because of this, Google was accused of being racist. This highlights how the culture in which algorithms are created matters: Google's algorithm was created by humans in a for-profit organisation geared towards connecting sellers and buyers of stock photography – a majority of which are White – in a capitalist society geared towards profit-making. Google's algorithm didn't understand that its commercial projectivity could be considered racist, nor did it have the reflexive and evaluative capacities needed to avoid the toxic feedback loop of search results.[2] Google has often insisted on the neutrality and objectivity of its algorithms, and that it is the users' searches that result in racist outcomes, and thus Google is not to blame. Indeed, as critical tech scholar Evgeny Morozov (2013) argues, Google's reluctance to acknowledge that its algorithms can malfunction allows it to extricate itself from a number of ethical aspects of its work.

1. Something Cukier and Mayer-Schoenberger warned against already in 2013 (see also Benjamin, 2019).
2. I have written together with Ulrike Klinger about how algorithmic agency differs from human agency along these lines (see Klinger & Svensson, 2018).

However, it is highly questionable whether it's possible to blame artifacts for mistakes in their design. In the bestseller book *Algorithms of Oppression: How Search Engines Reinforce Racism*, communication scholar Safiya Umoja Noble (2018) argues that it is the combination of private interests in promoting certain sites, along with the monopoly status of a relatively small number of Internet search engines, that leads to biased search algorithms which privilege whiteness and discriminate against people of colour.[3] Indeed, algorithms and automated systems operate with biases just like the rest of society; designed and engineered by humans, mistakes and unintended consequences are likely. In other words, to understand the prejudices of algorithms and automated systems, it's important to direct attention to the people involved in their design and engineering.

Social media take on a particular role here, as they are increasingly replacing traditional media channels as information intermediaries. Facebook is the number-one source of news about government and politics for a majority of so-called millennials (see Diakopoulus, 2016). By sorting, filtering, and ranking information, social media platforms direct attention towards some ideas, goods, and services, and away from others. Twitter is another example. If a tweet becomes a trend, it attracts more attention, following a so-called popularity principle (see van Dijck, 2013). Hence social media algorithms are not a reflection of reality – they *create* the reality by shaping public interest. As Pasquale (2015) argues, social media algorithms profoundly influence decisions about what to do, what to think, and what to buy. They produce the world they claim to merely show. In journalism studies, it has been known for a long time that news media not only report the news, they *make* the news. But, if you compare the number of studies on journalists shaping our imaginations and worldviews – often with arguments about their power in democracies – with the number of studies on programmers, it becomes apparent that we need to know more about the latter group.

Reading about social media biases, racism of search engines, and feedback loops in automated systems, it is easy to become disillusioned. Many researchers have also painted a gloomy picture (see, e.g., Chang, 2018; Morozov, 2011; O'Neill, 2016; Tufekci, 2015; Wachter-Boettcher, 2017; Zuboff, 2019). At the same time, it is important to not forget that these systems in many cases make our lives easier, more fun, and open up opportunities for increased connectivity among friends and family. Users can tailor services and experiences to their individual needs and desires. I, for example, enjoy these systems, as they allow me to be lazy and entertained and make it possible for me to spend my time commuting to work in a more fun or purposeful way.

3. In the history of photography, whiteness was early established as a norm (Sandvig et al., 2016), and this has embedded photography in racist assumptions that, 150 years later, account for the difficulty of non-White people to even be recognised as humans by algorithms. An infamous example of this is when face-detection algorithms suggested that Black people were gorillas, or consider the example of a digital camera suggesting that a smiling Chinese woman was blinking and the picture should be retaken.

Furthermore, during my journey into tech culture, I found that unfair results of algorithmic systems are most often unintended, often due to a practice of rolling out software a little at a time, getting feedback, and correcting mistakes along the way (as I discuss in more detail in Chapter 6). For example, Ted, a middle-aged app programmer I interviewed in my hometown, Malmö, described how he and his colleagues were discussing the creation of a dating app with a location-finder function for the user's potential date or love interest. It was not until a female acquaintance told them they were on the brink of creating a stalking app that the all-male group realised its potential for negative use. When it was brought to their attention, they were quick to dismantle the entire project.

There will always be unforeseen consequences of any system; programmers cannot anticipate every kind of use or every future circumstance their system will be in. As philosopher Bruno Latour allegedly said, unpredictable consequences are the most expected on earth (cited in Morozov, 2013: 337). Creating end-user application software means anticipating myriad combinations of human actions and machine responses. However, programmers might not be the best equipped for such anticipation, since their gut feelings may not be widely shared by the general public, as journalist Scott Rosenberg (2008) argues. Bart, a Dutch programmer who was travelling Europe to give talks, and whom I met in Copenhagen, underlined this; according to him, it's "incredibly difficult to leave assumptions behind when thinking about the users". Even if you could think about all the ways in which users could use the product, it's "so easy to forget something or to think to yourself that it's obvious to click on that button", while the user might have no clue or "find it obvious to *not* click on that button". Indeed, programmers are not the general public.

This book is about my journey into tech culture and my attempt to get to know the humans behind our screens – the programmers – and their cultures, from my perspective in media, communication, and critical data studies.

Who is this book for and why should you read it?

This book is about programmers and their cultures and the logics of programming. For the most part, it is an empirical and academic study, but it is also a personal account of a two-and-a-half-year journey into tech culture – going to conferences, interviewing programmers, and even learning to code somewhat myself in the process. Such an empirical backdrop has the potential to greatly inform discussions of the systems of power that programmers mobilise and that contemporary connected societies so much depend on; however, I leave the larger critical analysis aside until the final chapter.

Data, algorithms, and automated systems have also been critically analysed by others in meritorious ways. For example, media sociologist Helen Kennedy

(2016) wrote about data-mining and how it leads to new forms of data relations, and its consequences. She claims that few can access data, and even fewer can process it, while users of digital systems are all part of its production. Paraphrasing Karl Marx, workers haven't gained ownership of production, but have lost ownership of consumption through the exploitation of the data traces they leave behind. Social psychologist Shoshana Zuboff (2019) similarly argues that knowledge is extracted *from* us, but not *for* us, and how this data-mining not only predicts but also modifies our behaviours in the imperative of what she labels surveillance capitalism. Google co-founders Sergey Brin and Lawrence Page (1998: Appendix A), when they presented the Google large-scale search engine prototype in 1998, even admitted that "we expect advertising funded search engines will be inherently biased towards the advertisers and away from the needs of the consumers". Critical tech scholar Sara Wachter-Boettcher (2017) therefore urges us to inquire into who is being served by digital media and who is left behind, alienated, or insulted. Authors such as O'Neill (2016), Eubanks (2017), and Noble (2018) have shown that poor women of colour are particularly at risk here.

While I am hoping this book will be informative for those interested in a larger ideological critique of data and algorithms, it is a microlevel rather than a macrolevel study, as it is based on interviews with individual programmers. The larger critical discussion of shifting relations of power in contemporary societies permeated by data, digital media, and automated systems is merely my point of departure, justifying my focus on the humans behind the screens. Hence, while this book is partly directed at those interested in contemporary connected societies and shifting relations of power, it is also my aim to make this book accessible for a general public interested in knowing more about programmers and their cultures, and perhaps also for programmers themselves, who may be curious about an outsider's view on their culture(s) and practices.

There have been many calls for empirical studies of programmers. Media theorist Friedrich Kittler apparently labelled programmers engineers of being, treating them (rather than philosophers) as history's most important actors (in Peters, 2015). Media scholar Robin Mansell (2012) similarly wrote that citizens rely on programmers to ensure our communication system is consistent with aspirations for a good society. Rosenberg argued already in 2008 that people depend on software, while the art of making it remains a mystery: "never in the history of time have we depended so completely on a product that so few know" (Rosenberg, 2008: 9). Programmer and science fiction writer Vikram Chandra (2013: 1) seems to agree when he claims that while computing has transformed all our lives, "the processes and cultures that produce software remain largely opaque, alien and unknown". Algorithms and automated systems are works of collective authorship – made, maintained, and revised by many people with different goals at different times (see Seaver, 2013). In other

words, it's important to study how programmers' rules, ideals, and imaginations relate to the larger cultural setting in which data is mined and algorithms and automated systems are engineered.

The list of researchers, writers, and journalists emphasising the importance of programmers can be continued. Computer scientist and venture capitalist Paul Graham (2010: ix) argued that "if you want to understand where we are, and where we're going, it helps if you understand what's going on inside our heads". Computer journalist John Markoff (2015: xvii) wrote that "the best way to answer questions about control in a world full of smart machines is by understanding the values of those who are actually building them", underlining the scant discussion about programmers from this perspective. Similarly, Tristan Harris (2017) – the Google drop-out and founder of the nonprofit Center for Humane Technology – proclaimed in a TED Talk that a handful of people in a handful of companies are shaping the thoughts and feelings of billions of people and that it's important to understand these people. Ethics scholar Engin Bozdag pointed out already in 2013 that algorithms were influenced by humans. More recently, media scholar Taina Bucher (2018) claimed that human decision-making processes and programming precedes any algorithmic operation, and media researcher Jonas Andersson Schwarz (2019) emphasised that our digital media experiences are conditioned by those who design its structural set-up.

In other words, many – and from diverse fields – have underlined the importance of studies addressing the people and cultures behind data and algorithms. Fewer have actually conducted such studies. There are exceptions,[4] but there is definitely room for more studies. To illustrate this point, when preparing for my research grant application to the Swedish Research Council (in 2017), I booked a librarian at my university library in Uppsala, where I was employed at the time. Her task was to find studies of people behind data, algorithms, and automated systems. Since she had a master's degree in finding information and knew about most databases and library catalogues, I knew she would be much more competent than me to find such studies. To her great frustration, not much was found. She even involved her colleagues, as she had a hard time believing there were so few empirical studies. At this moment, I knew I was on to something.

This book addresses what many have called for but what few have actually conducted: empirical studies on programmers and their cultures. With increas-

4. For instance, Levy's (1984) study of hackers; Rosenberg's (2008) following a group of software developers during three years; Kunda's (2006) ethnography of a high tech corporation; Turner's (2006) study of Stewart Brand and the Whole Earth Network; Ensmenger's (2012) history of computer software that made the computer revolution; Markoff's (2015) account of the schism between AI and IA; Ananny and Crawford's (2015) study of news app developers; and Gillespie's (2018) and Roberts's (2019) studies of content moderation, just to mention a few.

ing amounts of data and computers with processing power, algorithms and automated systems will probably exercise even more power over us and our societies in the future, and it is unlikely that this power will diminish any time soon. An empirical study of the people and cultures behind data, algorithms, and automated systems is thus both important and timely.

Outline

In Chapter 2, I situate the study, further justify the focus on programmers from an existential media studies perspective, account for my point of departure in media and communication studies, and provide some details on how the empirical data was gathered and analysed using culture as my main lens to study and understand programmers.

The rationale for Chapter 3 is to illustrate that tech culture has many influences and is not a monoculture. It has roots in both hacker and hippie culture and in both the military-industrial complex and the 1960s counterculture. It is capitalist, libertarian, and entrepreneurial, as well as progressive and bohemian. At the same time, tech culture oozes masculine adolescence. It is playful and prankish with its emergence in anti-authoritarian suburban middle-class Silicon Valley garages. This chapter thus highlights some of the history of tech culture, Silicon Valley, and the move from the east to the west coast of the US.

As tech culture has actually sprung from many different cultures, it is important to underline the contradictions in order to complexify and gain a more sophisticated, and arguably more accurate, understanding of the culture and its people. In Chapter 4, I therefore discuss how community and sharing are considered important, while lone wolves are, at the same time, admired. Programmers take pride in their code, while simultaneously expected to work invisibly behind the screens. The underdog is a hero, while programmers today are conceived of – and treated – as an elite (as I will exemplify in the next chapter when I describe flying into Austin and SXSW).

The subsequent chapters deal with the various aspects of tech culture outlined in Chapter 2. In Chapter 5, I direct attention to symbols, labels, and heroes. I discuss what the bean bag, Lego, and unicorns actually symbolise and why the underdog, cowboy coder, and prankster are admired in the culture. The topic for Chapter 6 is the rules of programming: programming routines and rituals and ideals of best practices such as flow, grit, tinkering, and data-driven development. In Chapter 7, I examine the values and imaginations in the culture, such as a belief in data, progress, and a future that can be changed through technology.

In Chapter 8, I discuss the many references to magic in tech culture and how programmers make magic through manipulating code. I then summarise my findings and present the main finding of my journey into tech culture, namely,

the centrality of magic and that programmers can be labelled wizards – of a more benevolent kind of magician based on mastery of programming languages and imaginations of a better tomorrow.

Finally, in Chapter 9, I provide a summary of the book and make an outlook with the help of the concept of mathemagics. Here, I rely on sixteenth-century philosopher and theologian Giordano Bruno's tripartite delineation of magic, of which mathematical magic is one. Bruno also warned against mathematical magic becoming evil, something that resonates in contemporary critical data studies. Hence, I return to the larger critique of contemporary data societies that I depart from in this introduction, but through the lens of Giordano Bruno and mathemagics. I also connect mathemagics to Modernity and argue that tech culture is deeply modern. As a contrast, I suggest that death – that is, having a body that decays and eventually dies – is what makes life magical. I conclude the book by underlining the importance of democratising tech imaginations.

Chapter 2

Situating the study

I approach this study of the people behind data, algorithms, and automated systems from a background in media sociology and my departure point in critical data studies. In this chapter, I introduce myself so you can get to know me and where I come from. I follow this with an account of what I already knew about programmers before embarking on my journey. Lastly, I discuss my empirical data, how I collected it, and with what analytical lenses I approached programmers and their cultures.

The importance of digital technology in contemporary data societies

As a Swedish media and communication scholar, I belong in the social sciences; this has a bearing on how I understand digital technology, media, culture, and society at large. In my academic tradition, we consider society and its development as deeply intertwined with media innovation. So-called mass society was accompanied by the advent of mass media (see Curran & Gurevitch, 2005), and with the rise of network media, accounts of network societies became prevalent (see Castells, 2000; van Dijk, 2006). Today, with the rise of social media and datafication,[1] it is possible to label our highly connected societies as data societies – that is, societies characterised by digital technologies and datafication of basically everything. Communication is indeed fundamental for our cultures, civilizations, and the human species in general. In cognition studies, some even claim that our brain developed (and we humans became Homo Sapiens – "the knowing man") when we developed a language and began communicating more complex and abstract matters (as in McCrone's book from 1990, *The Ape that Spoke*). Historian Yuval Noah Harari (2014) argues in his bestseller, *Sapiens: A Brief History of Humankind*, that it was language which made it possible for our species to conquer the world.[2] In other words, media and communication is connected to our very existence, social as well as biological.

1. The term datafication underlines that almost all aspects of our lives are rendered into data, used for various algorithmic calculations (see Cukier & Mayer-Schoenberger, 2013).
2. With a more sophisticated language came the possibility of large-scale cooperation, imagination, and social coordination.

This is perhaps most explicit in the area of existential media studies. Media historian John Durham Peters (2015: 15) argues that media are both "the habitats and materials through which we act and are". Media is not only about the world – media *are* the world. To discuss digital media thus becomes equivalent to asking what existence is, as digital media are becoming environmental, the background of life, and our infrastructure of being. As philosophers Ciano Aydin, Margoth González Woge, and Peter-Paul Verbeek (2019: 337) write:

> Technology is becoming a mediating milieu, merging with the world to the point of becoming invisible, but at the same time intentionally directed at humans and helping to shape how humans act, perceive, and live their lives.

Hence, an instrumental view of media and communication technologies as outside humanity is refuted. Instead, users emerge through – or in tandem with – the tools (i.e., instruments) they use. As media scholar Amanda Lagerkvist (2019: 1) puts it, today's environmental and wearable, all-encompassing and increasingly automated technologies "co-shape, bring about and transform the human condition". Scholar Sun-ha Hong (2019) argues that knowing (as a mode to relate to the world "out there") is being reconfigured through digital media. He exemplifies this with the Quantified Self movement, a community of experimenters in self-tracking technologies who hope that, through smarter machines and more intimate and persistent measuring, they will reach a higher degree of self-knowing. That the machine knows more about us than we do ourselves is nicely illustrated by an anonymous Facebook user: "I'm never quite sure if Facebook's advertising algorithms know nothing about me, or more than I can admit to myself" (cited in Andersson Schwarz, 2019: 69). In other words, measuring technologies have become ingrained in the experience of the self. As Chris Dancy, whom I first met in 2018 at the Öredev (Öresund developers) conference in Malmö, proposes, home is where our sensors are. Chris is, according to himself, the most connected man on earth: a mindful cyborg and techno-pagan. In his very personal book from 2018, he writes that if, for some reason, his Fitbit[3] would fail to count his steps, it would be as if these steps had never been executed. Dancy thus underlines that it's not possible to escape technology, because users *become* technology.

Using an app to keep track of my calorie intake and burn, I recognise this behaviour. However, instead of failing to count my exercise activities (these I register meticulously), I sometimes cheat on the amount of ice cream and crisps I register – because if it isn't measured, it doesn't exist (right?). I guess I'm an example of what Evgeny Morozov (2013) critically labels a datasexual: completely absorbed with my personal data. In data societies, it's therefore

3. A Fitbit is an activity tracker, a wireless-enabled wearable technology device that measures data such as the number of steps walked or climbed, heart rate, quality of sleep, and other personal metrics involved in fitness.

possible to update philosopher René Descartes' famous proposition, "I think, therefore I am", with "I am measured (rendered into a quantitative format), therefore I am".

Existential media studies follow in a tradition of conceiving media technologies as extensions of ourselves.[4] This could be a reason why I feel lost without my phone, feeling so-called nomophobia – the irrational fear of being without your mobile phone or being unable to use your phone. My students aren't any better than me (not in this domain at least). In my role as a teacher, I often ask first-year students to keep a diary of their media use and related reflections. A common theme during my ten years conducting such media diaries is the fear of running out of battery power. As with Dancy's Fitbit, happenings – such as the first-week freshmen initiation activities – wouldn't have happened if not documented by pictures, uploaded to social media platforms, and generating likes from peers. I have myself written about this urge to be updated, in a double sense (see Svensson, 2012, 2015). The double nature of updating is about being updated on your network, as well as updating your network with idealised and reflexive taglines and pictures of yourself and your exciting everyday life. Indeed, social media users need others to *confirm* their reflexive self-projections. Thus, perhaps the new version of Descartes' proposition should be, "I am updated (and acknowledged for these updates), therefore I am".

In previous work, I have elaborated on the concept of digital late modernity, underlining how late modern acceleration of reflexivity and individualism is accompanied by the rise of social media practices, and how this favours a more expressive type of rationality (see Svensson, 2011, 2015). One of the first to point this out was sociologist and psychologist Sherry Turkle, who wrote in 1995 about the expressive affordances of early MUD (multi-user domains) – how users became authors not only of text, but also of themselves.

Hence, there is no doubt that digital media have become part of users' existence with their inevitable quest for meaning and value. It has become "sticky", designed to affectively attach themselves to users and their sense of meaning and being. The Internet has become an intimate technology that touches upon every facet of our lives in data societies. It is impossible to avoid digital media, as we are *thrown* into a digital human condition in which our existence cannot be escaped, not even after death (see Lagerkvist, 2019). But while digital media limit users, they also open up possibilities within their limits. Hence, that our digital existence is co-constituted by technology is not the same as being *determined* by technology. At the same time, it's important to underline that some are more determined and surveilled by digital media and automated

4. Most notably in this regard is Marshall McLuhan in his book from 1964, *Understanding Media: The Extensions of Man*, but also in later cyborg studies (of which Donna Haraway's *Cyborg Manifesto* from 1985 is perhaps the most well-known).

systems than others, as studies of predictive policing (referred to in Chapter 1) have clearly shown.

At SXSW2019 in Austin, I listened to media theorist Douglas Rushkoff when he launched his book, *Team Human* (2019). In his presentation, Rushkoff said that software is like a language: English is a software, and "being on" English shapes our imagination. Dancy (2018) also talks about technologies as things we think *with* and that these things influence our thinking. The Netflix documentary series *Explained* (Klein et al., 2018–present) even argues that code exerts a profound regulatory effect on our lives, which the law could only have hoped for. Much like city planners decide how to live in a city, code decides how people live their lives online; people are represented, targeted, and regulated by data.[5] Users' online and offline existence cannot be separated any longer. As Peters (2015) writes, if Google can't find you, you don't exist. This in turn resonates with Friedrich Kittler's claim that only that which is switchable exists – that wiring precedes being. In other words, data and digital media have become existential. As digital technologies co-constitute our existence and are part of our very being, programmers and their cultures are thus pivotal to study.

Who are the programmers?

While largely unknown, programmers are highly recognised and cheered in contemporary data societies. Christopher Steiner (2012) argues that the people who are – and will remain – the most indispensable in this economy are those who can build, maintain, and improve upon code and algorithms. This is also exemplified in Scott Rosenberg's (2008: 31) in-depth account of a development software team whose staff meeting was interrupted by former Vice President Al Gore (!) cheering the team: "Keep up the great work, you guys are changing the world".

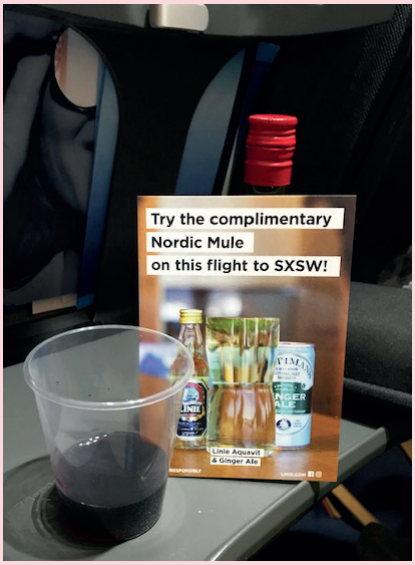
Programmers as rock stars

Travelling to SXSW2019 in Austin, I witness first-hand the importance that is projected on tech – and the people that populate it. As a media and communication researcher – having to fight for every penny for my research – I'm mind-blown (and slightly provoked) by all the money being spent on the SXSW attendants. I'm acutely aware this is not a normal flight. Scandinavian Airlines (SAS) (who normally does not fly into Austin) scheduled a direct flight from Copenhagen in connection with SXSW2019 – and I'm lucky to have a seat on it. A DJ is playing

5. This to the extent that Cheney-Lippold (2017) claims that our citizenship becomes dependent on what we do online (jus algorithmi) rather than on our parents (jus sanguini) or where we were born (jus soli). Hence, he proposes his own update of Descartes' proposition: "My data is seen, therefore I am".

funky music at the gate, and the Danish microbrewery Mikkeller has given each passenger a goodie bag with four types of beer (not that I need any extra alcohol, since this flight has free access to the bar – even for me seated in economy). Norwegian aquavit company Linie is also launching a new drink: the Nordic Mule (see Figure 2.1). All passengers are invited to try it out.

Figure 2.1 *Flying high with SAS*



Source: photo by Jakob Svensson

Instead of waiting idly at the departure gate, I test virtual reality (VR) and augmented reality (AR) simulations. The CEOs of SAS and Copenhagen Airport give short speeches before the boarding, and both underline the importance of SXSW and express hope that we passengers will bring inspiration and innovations from Austin back home to Scandinavia. We are cordially invited to the formal opening of House of Scandinavia (a pavilion in Austin, of which SAS is the main sponsor and in which events host prominent guests such as Crown Princess Mary of Denmark and ABBA legend Björn Ulvaeus). To top it all off, in the air somewhere above Canada, we're more or less forced to be part of the world premiere of the song "Good for Me" by Swedish artist Paul Rey (which can be viewed on YouTube, see Paul Rey, 2019). Slightly tipsy from all the free alcohol, I wave my glow sticks (provided by the overly cheerful crew in a dimmed cabin) and enthusiastically clap my hands together with SAS CEO Richard Gustafsson, who has joined us in economy from his business-class seat.

In Austin, following the House of Scandinavia launch, we're bestowed with even more Danish beer and food from Swedish celebrity chef Paul Svensson,

who, together with colleagues from Denmark, Norway, and Iceland, has transformed local food waste into gourmet meals under the label sustainable haute cuisine. I write in my research diary that tech culture and its people are afforded great importance, and to be seen and noticed in this culture seems pivotal. This is confirmed when I attend the all-inclusive SXSW festival, where I hop from the European Union pavilion, via House of Brazil and the LinkedIn Lounge, to the Sony Showroom, drinking free beer and snacking away at the expense of others.

I write in my diary that I feel like I'm among rock stars, with free-flowing alcohol, playfulness and partying, and a mashup tech conference, art exhibit, and music, film, and comedy festival. Being accustomed to academic conferences, this whole experience amazes me and underlines how much younger, hipper, more important – and harder to amaze – tech people seem to be compared with media and communication researchers like me.

What did I already know about programmers before embarking on this journey into tech culture? Well, they aren't your average Joe (or your average Svensson, as we say in Sweden). As one of my interview participants stated, "regular code is a recipe that's handcrafted by some human, armed with an opinion and a caffeine source".

In his study of a high-tech company (one of the few ethnographies of tech companies out there), organisation studies scholar Gideon Kunda (2006) describes a homogeneous group of White men in their late 20s to mid 30s. And it seems this hasn't changed since he conducted his research in the 1980s. Programmers in general *are* young and most often White and male. Women and Black people are underrepresented, and in Silicon Valley, the focus is on young people, from whom it is thought most new ideas will emerge (see Fisher, 2018). Scientist and inventor James West states in a Ted Talk that minority groups in the US constitute 26 per cent of the population, but only 5 per cent of tech, and the numbers for women are 51 per cent of the population, but only 20 per cent of tech (TEDx Talks, 2015). Journalist Emily Chang (2018) writes that in 2017, Google employed 31 per cent women overall and 21 per cent in vital tech roles; the numbers for Facebook were 35 per cent overall and 20 per cent in vital tech. Indeed, Silicon Valley is *incredibly* White and male, as editor Jay Yarow (2015) argues. At SXSW2019, I noticed that Black women in tech even found it necessary to call for their own gatherings to support each other – perhaps not so surprising, as Chang (2018) concludes that Black women only hold 3 per cent of all computing jobs (and the number for Latina women is a mere 1%).

However, when walking around Silicon Valley and the tech headquarters during my extended visit to the US after SXSW2019, I actually saw quite a few women and also some East Asians (mainly Chinese) and South Asians. I hardly

saw any Black or old people though, and Latinos were mostly working in service functions such as at restaurants, as security, or in parking areas. This observation is verified by digital media scholar John Cheney-Lippold (2017), who claims that Silicon Valley's largest companies employ only 6 per cent Hispanic and 4 per cent Black people. Similarly, Frank Pasquale (2015) states that only 2 per cent of Google employees are African American, compared with 12 per cent of the American workforce. The latest numbers I could find for Sweden are from 2017, where a survey found that 16 per cent of the male population (16–85 years old) had written code in some kind of programming language, while only 5 per cent of the female population had done so (SCB, 2017). Another survey concluded that the number of women in IT and Telecom in Sweden in 2019 was 29 per cent (IT&Telekomföretagen, 2019).

When talking about biased systems, maybe the composition of the workforce is one reason. Face-recognition algorithms would probably recognise Black people if there were more Black people involved in their design, and image-search algorithms would probably be less gender-biased if more women were involved – much like the anecdote in Chapter 1 about the female colleague providing a different perspective to the dating (stalking) app designed by an all-male team. As Nivia Henry, head of ad experience at Spotify – and also a Black woman – stated in a talk at the Öredev2018 conference in Malmö, "representation matters, we have entered the age of automation overconfident, yet underprepared".

Someone who has been particularly critical of the tech industry in this regard is Sara Wachter-Boettcher (2017), who has written a book with a telling title, *Technically Wrong: Sexist Apps, Biased Algorithms, and other Threats of Toxic Tech*. She describes a culture that routinely excludes anyone who isn't young, White, and male: a culture populated by White guys who, from young age, have been told they are the best, and hence wholeheartedly embrace the idea that they are truly smarter than everyone else and deserve to make choices on their behalf. In the same vein as Wachter-Boettcher, Chang (2018) writes critically in her book, *Brotopia: Breaking up the Boy's Club of Silicon Valley*, about how women face toxic workplaces, marked by discrimination and sexual harassment.[6]

There seems to be an awareness of this young-White-male bias, as tech companies purposely target women and minority groups. For example, attending Öredev2018 in Malmö, I noticed the gender-neutral toilets marked with a picture of a rainbow-coloured mermaid and centaur (see Figure 2.2). In recounting my journey into tech culture, I try to provide nuance for some of these critical accounts. But before continuing this journey, I should attend to how I went about studying programmers and their cultures.

6. Even though I'm quite sceptical of such black-and-white accounts, I do agree with Wachter-Boettcher when she argues that it is important to ask who decides what, with what desired outcome, on basis of what data, and how good or fair results are defined.

Figure 2.2 *Gender-neutral and whimsical, Öredev2018*

Source: photo by Jakob Svensson

Empirical data gathering

For the most part, this book draws on an interview study targeting software engineers, algorithm programmers, and developers, coupled with observations of various tech conferences and meetups. But I also conducted an in-depth study of a particular algorithm in a particular setting: the process of introducing and implementing an algorithm to rank and mix news on the front page of a leading Scandinavian daily newspaper (henceforth, the Daily). The reason for this study was *not* to contribute to the rather burgeoning field of digital journalism, but to complement an interview study of disparate programmers in different countries by also studying programmers in a specific context.

Focusing on interviews and observation, the method used is best labelled *ethnographically inspired*. Being an outsider to tech culture also resonates in early ethnographic accounts of mainly Western anthropologists travelling to faraway places to study and understand unknown cultures (see, e.g., Evans-Pritchard, 1937; Malinowski, 1922; Mead, 1928). Since the 1970s, there has been a push away from the unknown and faraway to the known and nearby (see Hylland Eriksen & Nielsen, 2004). But this study is different, in

that the culture I set out to explore is not completely unknown, as our very existence in connected data societies is conflated with digital media, as well as an outcome of this culture. My everyday environment is permeated with services and products programmed, designed, and engineered within tech culture; hence, I'm *both* an insider (in that I am a skilled tech user), but also an outsider (in that I know very little about the culture that has formed these technologies). This opens up opportunities – as well as problems – in my quest to understand the culture.

My strategy has been to adopt the role of a tech virgin and not take any answers for granted, in order to push interview participants to really delve into what might be seen as self-evident in order to really get into the heart of the culture. I also pushed myself to leave any preconceptions behind as I observed tech conferences, trying to take note of the self-evident and mundane as new and alien experiences. In this sense, my observations were not fully participatory, as I did not attempt to become part of the culture and live life as a programmer. However, being genuinely interested in the art and craft of programming, I did learn to code some during this process in addition to attending some meetups.

I began my empirical data collection by re-connecting with an old friend who is a freelance programmer. I visited him in his hometown to share my ideas for the project and to get a better understanding of programmers and their work. From there, he "snowballed" me towards another friend, a programmer who happened to work at the Daily. I invited him for lunch to talk about algorithms, his work, and my ideas for this project. He was very helpful and interested and put me in contact with a manager at the Daily. After some e-mailing back and forth, they agreed to be part of the study.

For the larger interview study, I adopted a mixed recruiting strategy: snowballing personal contacts and approaching programmers through the platforms LinkedIn and Meetup.[7] A colleague of mine who had conducted interviews in Silicon Valley confirmed it was relatively easy to be granted time with programmers and software engineers and advised me to get a LinkedIn premium account and contact potential participants through it. This was a good investment, as I could filter programmers according to their location and particular work tasks. For example, before going to Silicon Valley, I used the search function to find programmers working at the big social media companies Facebook, Google, and LinkedIn in the Bay Area (the region including San Francisco and the surrounding counties). In Scandinavia, I also homed in on programmers working in social media companies with offices in

7. LinkedIn is a business- and employment-oriented service mainly used for professional networking, such as employers posting jobs and job seekers posting their CVs. Meetup is a service used to organise online groups that host in-person events for people with similar interests. Both services seem to be very popular in the programming community.

Copenhagen and Stockholm, sending them a so-called in-message, in which I tried to make it clear that my focus was on them, and not the company they work for.

On Meetup, I looked for events in the cities where I would be conducting interviews or attending conferences. I also wanted to target certain groups in particular, such as "Women who code Silicon Valley", in order to diversify my sample. In addition, at times I approached programmers informally at conferences if I felt they could be beneficial for the diversity of my sample. Table 2.1 lists the formal interviews I conducted.

Table 2.1 *Interviews conducted*

Name	Where	Nationality	When	Age (gender)	Occupation
Niklas	Norrköping	Swedish	Feb 2018	late 30s (m)	programmer/freelance
Pelle	Stockholm	Swedish	Feb 2018	late 30s (m)	programmer/freelance
Andri	Malmö	Estonian	March 2018	late 30s (m)	programmer/company
Sören	Copenhagen	Danish	March 2018	40s (m)	programmer/tech giant
Ted	Malmö	Swedish	March 2018	40s (m)	programmer/own startup
Bart	Copenhagen	Dutch	April 2018	30s (m)	programmer/freelance
Gabriel	Stockholm	Swedish	April 2018	early 30s (m)	web development/Daily
Jonas	Stockholm	Swedish	April 2018	early 30s (m)	web development/Daily
Per	Stockholm	Swedish	April 2018	late 40s (m)	web development/Daily
Anders	Stockholm	Swedish	April 2018	late 30s (m)	programmer/Daily
Daniel	Stockholm	Swedish	April 2018	30s (m)	programmer/Daily
Adam	Lund	Swedish	May 2018	30s (m)	programmer/freelance
Viktoria	Stockholm	Swedish	Sep 2018	40s (f)	UX designer/company
Stina	Stockholm	Swedish	Sep 2018	late 20s (f)	social media editor/Daily
Viktor	Stockholm	Swedish	Sep 2018	30s (m)	programmer/Daily
Douglas	Stockholm	Swedish	Sep 2018	50s (m)	programmer/company
Nils	Stockholm	Swedish	Sep 2018	50s (m)	web development/Daily
Lasse	Stockholm	Swedish	Sep 2018	late 30s (m)	programmer/company
Jakob	Stockholm	Swedish	Sep 2018	30s (m)	programmer/Daily
Oskar	Stockholm	Swedish	Dec 2018	20s (m)	programmer/Daily
Python	Berlin	German	Jan 2019	40s (m)	programmer/freelance
Sunil	Berlin	Indian	Jan 2019	20s (m)	programmer/company
Benjamin	Skype	Israeli	Jan 2019	late 30s (m)	programmer/freelance
Kristina	Berlin	Polish	Jan 2019	early 40s (f)	programmer/student

Table 2.1 *Cont.*

Name	Where	Nationality	When	Age (gender)	Occupation
Chris	Austin	American	March 2019	early 50s (m)	writer/freelance
Pedro	Austin	Mexican	March 2019	early 30s (m)	programmer/freelance
Martin	Silicon Valley	Danish	March 2019	late 20s (m)	programmer/tech giant
Roger	Silicon Valley	American	March 2019	50s (m)	programmer/tech giant
Sam	Silicon Valley	Chinese	March 2019	20s (m)	programmer/tech giant
Nadja	Silicon Valley	Lithuanian	March 2019	50s (f)	programmer/own company
Anna	Silicon Valley	American	March 2019	50s (non-binary)	programmer/company/ NGO
Kim	San Francisco	American	March 2019	30s (trans)	programmer/freelance
Mark	Skype	American	April 2019	30s (m)	programmer/freelance/ NGO
Hans	Malmö	German	April 2019	40s (m)	programmer/freelance
Nilesh	Bangalore	Indian	Nov 2019	20s (m)	programmer/company
Vikas	Bangalore	Indian	Nov 2019	late 20s (m)	programmer/company
Narendra	Chennai	Indian	Dec 2019	late 20s (m)	programmer/company
Thiago	Sao Paolo	Brazilian	Feb 2020	30s (m)	programmer/freelance
Gabriela	Sao Paolo	Brazilian	Feb 2020	20s (f)	programmer/company

Comments: 39 formal interviews were conducted. All but one of the interviewees (Chris Dancy, a public figure) were given pseudonyms, and all but two interviews (Benjamin and Mark) were conducted face to face.

The qualitative interviews resulted in a wealth of data: over 35 hours of interview material, which took me almost 5 weeks to transcribe into over 450 pages.[8] The interviews took place at the companies where respondents worked, or at cafés or lunch restaurants, and a few were conducted via Skype from home or the AirBNB apartments where I stayed during my travels. I analysed the interview transcriptions vertically (as a whole, that is, as one entity) and horizontally (theme per theme across all interviews).

Apart from the interviews, I also attended conferences and meetups in Austin, Bangalore, Berlin, Chennai, Copenhagen, Malmö, Sao Paulo, Silicon Valley, and Stockholm. The Meetup platform allows one to search for events concerning a particular topic (such as coding) within a distance one sets themself from a location (for example, Sunnyvale, California, where I stayed during my Silicon Valley visit). Hence, the focus on Meetup was to find events that I could observe, but also to find interviewees to diversify my sample. Table 2.2 lists my formal observations.

8. Interview quotes originally in Swedish have been translated by me into English.

Table 2.2 *Observations*

Location	When	Duration	Context
Stockholm	Feb 2018	2 hours	meetup
Copenhagen	March 2018	2 hours	tech giant
Stockholm	April 2018	2 days	web development team at the Daily
Copenhagen	April 2018	2 hours	meetup
Malmö	May 2018	2 hours	meetup
Copenhagen	Aug 2018	2 hours	meetup
Stockholm	Sep 2018	3 days	data analytics and web development teams at the Daily
Copenhagen	Sep 2018	2 hours	meetup
Malmö	Nov 2018	1 hour	meetup
Malmö	Nov 2018	3 days	Öredev conference
Berlin	Nov 2018	1 day	React day conference
Berlin	Dec 2018	2 hours	meetup organised by tech giant
Berlin	Jan 2019	2 hours	meetup
Berlin	Jan 2019	2 hours	meetup
Copenhagen	March 2019	1 hour	gate entertainment before Austin flight
SAS flight	March 2019	10 hours	SAS direct flight to Austin
Austin	March 2019	10 days	SXSW conference and festival
Silicon Valley	March 2019	4 hours	tech giant headquarters
Silicon Valley	March 2019	2 hours	tech giant headquarters
Silicon Valley	March 2019	2 hours	Meetup
Silicon Valley	March 2019	2 hours	tech giant headquarters
Silicon Valley	March 2019	2 hours	meetup
Oakland	March 2019	2 hours	tech NGO/startup
Oakland	March 2019	3 hours	tech NGO/startup
Copenhagen	May 2019	2 hours	meetup
Copenhagen	Sep 2019	2 hours	meetup
Malmö	Sep 2019	2 hours	meetup
Berlin	Nov 2019	2 hours	meetup
Berlin	Nov 2019	1 day	React day conference
Bangalore	Nov 2019	2 hours	meetup
Bangalore	Nov 2019	4 hours	eCity
Bangalore	Nov 2019	4 hours	tech giant
Chennai	Dec 2019	2 hours	meetup
Copenhagen	Jan 2020	2 hours	meetup
Malmö	Feb 2020	1 hour	meetup
Sao Paolo	Feb 2020	1 hour	meetup

During the fieldwork – going to conferences, driving around Silicon Valley, visiting exhibitions, the Computer History Museum in Mountain View, and other museums, conducting interviews, and reading books – I kept a research diary and took over 500 photos, which proved useful when summarising my observations and impressions. Re-reading my almost 200-page-long diary reveals many impressions from outside these more formal observations and interviews, including from documentaries, podcasts, radio shows, television series, exhibitions, museums, and so on. I even have notes of my impressions from reading my university teacher union magazine, as well as Finnair's in-flight magazine. Thus, my arguments in this book aren't solely based on organised and formal empirical data gathering, but also from living my life during these almost three years, having these questions in my head, making connections, and actively choosing podcasts and documentaries when winding down after a full day of teaching. I don't believe in research as a detached practice. And even though I have tried to be critical (also of myself and my own assumptions), I haven't tried to eliminate myself from the research process, or from this book. Hence, I will not try to mask some of the more personal connections and revelations I have made during my journey.

Culture as an analytical tool

The interviews and observations were not random, but rather structured around a number of sensitising concepts, semistructured and also theoretically informed, drawn from analytical theories of culture and logics. As I develop further in what follows, culture can be approached in terms of symbols, heroes, practices and values, and logics in terms of rules, values, and imaginations. As such, the concepts of culture and logics intersect and aren't easy to separate. Chapters 5, 6, and 7 follow these sensitising concepts through which the culture is empirically and analytically approached.

My project draws upon tech anthropologist Nick Seaver's (2013) argument that algorithms should be understood as systems, and these systems aren't standalone little boxes, but massive networked ones, with sometimes hundreds of hands reaching into them. According to Seaver, it is important to examine the logics guiding these hands.

The concept of media logics was first theorised by now famous media sociologists David Altheide and Robert Snow (1979). In this project, I draw from my own and Ulrike Klinger's theory of network media logics (see Klinger & Svensson, 2015, 2016). We approach media logics as rules of the game of particular media, meaning the specific norms, values, and processes underlying how information is produced, distributed, and used. This can be further differentiated along lines of ideals, commercial imperatives, and technological affordances.

I have elsewhere operationalised the media logics framework around the sensitising concepts of rules, values, and imaginations (see Svensson, in review), which interact with, and inform, each other in a dynamic circuit. I'm inspired by media scholar Peter Dahlgren's (2009) analytical frame of civic cultures, a dynamic circuit of sensitising concepts to understand media and political engagement. Dahlgren argues that a sociological study of the humans, cultures, and logics behind algorithms and automated systems must depart from a holistic approach and account for how different aspects inform and feed off each other in the particular context or culture under study. Because, as Kunda (2006: 8) writes, culture is intrinsic, "a learned body of tradition that governs what one needs to know, think and feel in order to meet standards of membership". Hence, you can trace culture in public expressions, signs, and symbols, shared rules governing cognitive and affective aspects of membership, and the means whereby they are expressed.

In this sense, culture – following sociologist James G. March and political scientist Johan P. Olsen's (1984) widely cited outline (even though they talk about institutions, or "new institutionalism", rather than culture) – is not only about individual actors, but also about structures, rules, procedures, and practices that have a bearing on, or even constrain, the individual. In this study, the individual programmers lead me to the larger culture. As sociologist Anthony Giddens (1984) famously argues in his theory of Structuration, a culture – its creation, maintenance, and reproduction – is based both on larger structures as well as its agents. Structure and agency are mutually constitutive in the same sense that media and technology are co-constitutive of society and our existence. In other words, when using individual programmers as my study's departure, I understand them as acting both *in* and *through* a culture. This allows me to approach tech culture anthropologically and from a micro level. It is through the programmers themselves and their accounts – and my observations of their actions, practices, and symbols – that I make suggestions on how to understand tech culture at large.

Breaking culture into more manageable sensitising concepts was helpful in structuring the analysis and chapters in this book. To approach culture, I (following Dahlgren) observed and asked about programmers' background, identities, group belongings, and so on. I also found it beneficial to use social psychologist Geert Hofstede's (1991) analytical model for understanding values in a culture.[9] According to Hofstede, values are at the very core of a culture (as I discuss in Chapter 7) and can be discerned by studying more outer (and hence more easily observable) layers of cultural manifestations, such as symbols, he-

9. Hofstede is a controversial researcher, since he has generalised cultures, putting them into tables and schemata, in a way fixating and stereotyping nations and organisations (see Oyserman et al., 2002). Even if this criticism is justified, it is his model, not his results, I have used for approaching and analysing cultures.

roes (see Chapter 5) and rituals, or rules of the game (see Chapter 6). It is also important to keep in mind the constant flux of a culture as something always in the process of change (even if slow). Indeed, culture must be studied in context.

Rules are understood as norms and processes, rules of the game "which define specific constraints and opportunities for actors depending on their structural location" (Klein & Kleinman, 2002). In this study, rules have been discerned by observing and interviewing about how programming is conducted: Who instructs whom and according to what pattern? What are the different expectations when programming and writing code? What different capacities are involved when solving the problems identified that code and software could solve? How is the field of relations structured?

Ideals have been studied in terms of values guiding the work. Professional values at the Daily follow so-called news values (even if they are being changed with digital media; see Svensson, in review). On social media platforms, values rather revolve around so-called produsage, connectivity, virality, and reflexive sharing of information among peers and like-minded individuals. Media scholars Mike Ananny and Kate Crawford (2015) find that news-app designers are informed by an ideal of trying to satisfy (what they believe are) user demands – in other words, to be user friendly. In my study, values have been studied through observing and interviewing about what education and skills are deemed necessary, but also by asking who and what are considered good examples (i.e., heroes) and what success stories are being circulated and drawn upon.

The imaginations of programmers and software engineers inform the processes of programming and engineering algorithms and automated systems. The power of imaginations in shaping algorithmic calculations cannot be understated, and I return to this in the concluding section of the final chapter. How do programmers perceive the platforms, technologies, and data contexts in which their algorithms and automated systems are put to use? This is about perceived technological affordances (Gibson, 1977), how media technologies have characteristics that both enable and restrict actors in their production, processing, and presentation of content (Hjarvard, 2013). Social media platforms afford interactivity and push for constant updating; what affordances do programmers imagine for the platforms they design their software for? Tapping into the concept of imagined affordances (see Neff & Nafus, 2016), I observed and posed interview questions about how programmers perceive or imagine the technology they design for (that is, their imaginations).

This is how I understand and approach culture in this book: through symbols, heroes, rituals, practices, rules, norms, values, and imaginations. It is important to underline that these sensitising concepts are not separate – they intersect and inform each other. A qualitative study of the cultures informing the practices of designing and programming algorithms and automated systems must deal with the messiness of the social – and its intersections with the technological.

These sensitising concepts have provided a rough map that I brought with me to the sites of empirical data gathering, as themes to structure interviews and observations around, as well as a foundational structure for how to present tech culture in this book.

Delineations, limitations, & ethics

This book is neither about organisational culture nor organisational logics at big tech companies – or smaller startups (for this, see Kunda, 2006). By observing tech conferences and talking with programmers about their work, values, and backgrounds, my aim has been to paint a rather broad picture of people who have an increasing amount of power in contemporary data societies. A limitation of painting with a broad brush is that detail, as well as contradictions in the material, tend to end up in the background. For example, within larger tech companies, there are differences between management and coders, and there are also quite substantial differences between coders, developers, and project owners. Hence, even though I use the label programmer as an umbrella term for the programming, engineering, and developing professionals within tech, this does not mean that a software engineer is exactly the same as a coder or a programmer, or that the meanings of these titles resonate across contexts, countries, and locations. Programmers from India are not identical to programmers from Denmark; indeed, one of the major differences identified in my material has been between programmers from the so-called West and those from India and China.

Another major difference in my material is between freelance programmers and those working in big tech. While the main focus of the interviews was on the programmers themselves, rather than the company they work for or the projects they are involved in, it is important to bear in mind that programmers often work for private companies that follow a capitalist logic of capital accumulation. The people behind our computer screens work in companies and projects with predefined targets, and algorithms and automated systems are engineered to solve some kind of problem and reach some kind of target.[10]

Not all technology, however, is moulded by capitalist companies. Social innovation is a case in point; yet, even in charity, money is needed. Hence, it is also important to study funders, venture capitalists, angel investors, and their reasons for deciding what to invest and not invest in. As Kim, a transsexual programmer I interviewed in San Francisco, underlined, even if there is more diversity on the engineering level, as soon as you get out of that level, the people

10. It would indeed be interesting to look further behind the programmers themselves to explore what problems they are instructed to work on and how these problems are formulated: Who formulates the problems they are tasked to solve with their algorithms and automated systems? Who sets the targets they are supposed to reach? However, this is not the topic of this book.

who manage and invest in products and services are "White straight cis-gendered men for the most part". Hence, a call for diversity should also extend to the funders of tech projects: the people who get to decide what is developed at all.[11]

With my focus on what lies behind our immediate eyesight, this study has traits of the increasingly popular (in media studies) area of infrastructuralism (see Peters, 2015). However, if most studies of media infrastructures deal with towers, cables, and physical things (see Parks & Starosielski, 2015), my focus is rather on humans: where do they come from and how were they formed as programmers? It is possible to argue that this is about what could be labelled the human and cultural infrastructures of data and automated systems. The body is the ultimate infrastructure, and the delineations between man and milieu, media and environments, and body and machine are indeed difficult to uphold, as Peters (2015) argues with his theory of elemental media. I won't mind if someone labels this a study of the *human* infrastructures of contemporary data societies.

Methodologically, one limitation of an interview-based study is that it relies on reflective accounts from participants often asked to remember their education, first computer, and so on. Furthermore, since this sample "only" includes 39 programmers, observations at 4 conferences, no more than 20 meetups, and 8 workplaces, generalisability is hard to claim.

Having snowballed personal contacts, I suspect a bias towards gay and freelance programmers: gay, because of my own sexuality, and freelance, my informed guess is, because this actually reflects the population of programmers out there, especially in the West. I suspect my sample contains more homosexuals, Swedes, old programmers, programmers working in news organisations, and people of different ethnicities than the programming community in general; thus, I do not claim my sample to be representative of programmers in general. However, my sample is large and diverse enough to initiate a discussion about the people and cultures behind data, algorithms, and automated systems.

Throughout my data gathering, I was completely open with the purpose of my research and who I am. Participants (apart from Chris) have been anonymised as much as possible, considering they are in a specific field and from a Nordic orientation; for example, someone familiar with the Scandinavian ecology might be able to figure out which newspaper the Daily is. And, how many Danes in their late 20s work at a tech giant in Silicon Valley? Following journalism and public relations scholar Robert Kozinets (2011), this level of anonymisation falls into middle-masking, which I explained to all participants and secured informed consent. A confidentiality agreement with the Daily was also signed.[12]

11. As Chang (2018) shows, there are male predators in VC (venture capital), and in 2016, women-led companies only received 2 per cent of VC funding (a topic for another study, as the funding for projects my interviewees were working on was not part of my scope).
12. Surprisingly, I did not have to sign anything at any of the other companies I visited. This is most likely due to the fact that these visits took place in relation to also conducting interviews. Hence, having signed in as my interviewee's guest, this meant I was their responsibility during the visit.

▼▼▼

I began this chapter with an account of my own position in media sociology and critical data studies. Supported by my observations, I underlined the importance that is projected on programmers in contemporary data societies. I presented my interview participants and observation sites and acknowledged the potential bias in my sample due to snowballing personal contacts and searching for individual participants through LinkedIn. I ended the chapter with some details on the empirical data gathering and the analytical framework. In the next chapter, I examine the roots of tech culture and the important role of the history of programming. I also discuss influences such as the spiritual hippie, the secretive hacker, the entrepreneur, and the adolescent middle-class *bro*grammer.

Chapter 3

A culture of many cultures

In this chapter, I attend to contemporary tech culture's many roots and influences and how they have merged and intermingled. As argued in the previous chapter, programmers act both in and through culture, and while agents in this culture, the culture affords certain types of behaviour. Hence, it is appropriate to continue with a more historical outlook of tech culture.

Sara Wachter-Boettcher (2017) writes from a critical perspective that tech is an insular culture, a monoculture where insiders bond over a shared belief in their own brilliance. She has a point in that the culture is quite homogenous in terms of age, ethnicity, and gender; as discussed in the previous chapter, it's young and predominantly populated by men of Caucasian or Asian origin. However, in order to understand programmers and the logics informing their work, it is important to acknowledge that tech culture is influenced from many different directions. The hacker is an important figure, but so is the hippie. The leftist and countercultural origins of Silicon Valley and the birth of the personal computer (PC) still make their marks on culture and everyday life in the Valley. Certain aspects of these origins, such as freedom of information, individual empowerment, and realising the future through code, have been picked up by the right wing in politics, coupling a capitalist logic of making profit with entrepreneurship and startup values. Some of these themes can also be traced to the culture's youth orientation, teenage rebellion, and anti-authoritarian prankish dude culture with roots in, first the computer labs of established universities, and later in the garages of middle-class suburbia.

Hippie culture

When walking around the seminars and locations in Austin during SXSW2019, I found a Zen Den Room in one of the Hilton Hotels. Here, stressed-out conference goers could walk into a completely white room with bright lighting to reconnect with their inner selves and become "more mindful", as described in the conference programme. Mindfulness indeed seems to be a theme in tech culture; perhaps its roots in hippie culture can provide some insight.

A special issue of *Time* magazine from March 1995 was labelled *Welcome to Cyberspace* (Time, 1995). In the issue, the readers were told to forget about

anti-war protests, Woodstock, and even long hair; the real legacy of the 60s generation, it was argued, is the computer revolution. "We owe it all to the hippies", as computer pioneer Stewart Brand put it (in Turner, 2006: 103). The hippie origins of Silicon Valley are portrayed in the documentary series *The Silicon Valley Revolution* (Tenhaven, 2017). The first people who came to the Bay Area were generally on the left of the political spectrum: "All computer scientists in the 70s were left, ate whole grain rice, and wore sandals", one interviewee in the documentary stated. According to the documentary, the whole movement was due to a general dissatisfaction with big companies (most notably IBM and Microsoft) and their intolerance of people who were different.[1] The first episode of the documentary tells the story of how Steve Jobs went to India with a profound interest in Zen Buddhism, psychedelics, and hippie spirituality in general.

Early tech culture was liberal in its embrace of psychedelic drugs, and many of the computer pioneers were spiritually oriented, both in terms of their interest in Eastern mysticism and the use of LSD. This is detailed by communication historian Fred Turner in his highly informative book *From Counterculture to Cyberculture* (2006). Marijuana, Peyote, and LSD offered a chance to engage in an experience of togetherness in which electronic technology played an important role. Technology, together with psychedelics, made early pioneers imagine themselves as parts of a mystical community, unveiling fundamental links between all living things and the otherwise invisible energies that linked and governed the material world. So-called Trip Festivals were organised, bringing together "the beatnik derived San Francisco psychedelic scene and the multimedia technophilia of art troupes" (Turner, 2006: 66). Indeed, early tech was an unorthodox mixture of fascination with machines, drug use, and consciousness studies – a merger of the technological and the psychedelic. "As Detroit was to the car, the Bay area was to LSD", according to San Francisco-based writer Erik Davies (in Fisher, 2018: 7).

In Turner's (2006) account, the youth of the 1960s branched into two movements: the "new left", outward-looking and oriented towards political action; and the "new communalists", inward-looking and searching for consciousness via Zen Buddhism, beat poetry, psychedelic drugs, and not least technology. The new communalists turned away from political action and towards tech and the transformation of consciousness as the primary source of social change – the key to change wasn't in politics but in the minds of people. Here, mathematician and philosopher Norbert Wiener's (1948) influential theory of Cybernetics – as well as the notion of the globe as a single interlinked pattern of information – was influential and comforting, since it offered a possibility of global harmony. Human beings, the natural world, technological systems, and institutions were

1. The legendary move from the east to the west coast was allegedly due to a discontent with IBM, or what is described as an "extreme aversion against Microsoft" (Thomas, 2002: 88–89).

believed to be reflected in each other and examples of connected patterns. This was also associated with media theorist Marshall McLuhan (1964) and his influential (within media and communication studies) ideas of how electronic media links all of humanity into a single global village. The computer offered a possibility to move through life, "not in hierarchical bureaucratic towers, but as members of flexible temporary and culturally congenial tribes" (Turner, 2006: 238). Stereo gear, slide projectors, strobe lights, and LSD were believed to have the power to transform the human mindset and link humans together to the invisible vibes of others, and computers seemed to bring to life a countercultural dream of empowered and mindful individualism, collaborative community, and spiritual communion.[2]

The hippie generation also had the opportunity to dream. According to *The Silicon Valley Revolution* (Tenhaven, 2017), the baby boomers (people born between 1946 and 1964) grew up in a period of increasing affluence; hence, many felt safe to experiment, and the ones who left the east coast were comfortable enough to question their peers and embrace differences, even computer nerds. "The weird is ok, you need free-thinkers", one interviewee said.

This embrace of the outsider, of differences, is still a part of tech culture. "UX was the only place I felt I belong", Viktoria, a UX (user experience) designer I met in Stockholm, told me. "In tech, you need different viewpoints", she continued. And when asked to reflect on how she ended up in UX, she again underlined her being different. She wanted to make things – to *change* things – rather than just "explore boys and make-up". Embracing differences was also underlined by Per, the head of a small tech development team at the Daily, when he discussed his recruitment strategy with me. "There is enough of me in this team", Per said with a laugh, "I need someone who is different, can challenge me, us, in the team".

Anti-authoritarianism is important here. This is a theme that surfaced especially in my interviews with freelance programmers, who highlighted their freedom to leave a job if it didn't suit them or if they didn't get along with the client, as well as being in charge of their schedule and not having a boss. "I hate having a boss, seriously, I don't do well with authority", as Bart in Copenhagen phrased it. Similarly, Pelle – a middle-aged Swedish programmer I met in Stockholm – stated that it would be fun to work on one's own projects, "not being told what to do, but to do it on my own terms". Adam, a low-key programmer I interviewed in the south Swedish town of Lund, also dreamed of his own projects. And the list goes on; the urge to be free and independent was something that many of my interviewees returned to again and again.

2. It's possible to find traces of this in more recent narratives of computers as tools for empowerment and imaginary cosmopolitanism (Morozov, 2013). Indeed, to personalise the computer – taking what used to be property of big capital and making it everyone's property – was one of the big battles for early tech pioneers. "Computers are coming to the people and this is the best thing since psychedelics", as Stewart Brand asserted (in Turner, 2006: 117).

This anti-authoritarianism can be traced back to the early hippie pioneers who wanted freedom from being bossed around by large companies. Not only should programmers as individuals be free, information should also be free. Microsoft Windows was accused of hiding the workings of their systems. Macintosh, when it was introduced in 1984, positioned themselves against an image of an oppressive, old, dusty, and corporate IBM, the epitome of the bureaucratic world; computer technology should be made freely available so that programmers could realise their visions and share them with the world. This resonates in what has become known as the software flap (see Levy, 1984).[3]

What later became known as open source can also be traced back to this ethos of anti-authoritarianism and embrace of togetherness and sharing. The early days of Silicon Valley witnessed computer clubs in which enthusiasts met and exchanged code and programs with each other. A free university was established in Palo Alto, where one could choose among a plethora of subjects but also *create* a course if it didn't already exist. According to *The Silicon Valley Revolution* (Tenhaven, 2017), there was even a course on "being gentle". The computer was viewed as a fantasy amplifier; if programmers had an idea, they could realise it by writing a program. And in order to gain respect in the community, programmers had to openly and freely share their code. This was put in contrast to IBM, who arguably wanted to sell programs to make money. Today, Silicon Valley is all about business, and the culture has transformed with the influence of entrepreneurship and startup culture, as I attend to later in this chapter.

Hacker culture

Hackers and hacking have a huge symbolic value in contemporary tech culture. It is no coincidence that Facebook's headquarters in Menlo Park is located at 1 Hacker Way. When I visited in March 2019, I was blown away by the sheer scale of the place and the amount of services, exquisite art, and delicious food, without opening my wallet once. Not only are there environmentally conscious services like bike repair shops, but also an in-house smokery with Texan barbecue, and art workshops where workers and visitors could make, for example, their own pins and posters and have them printed to bring back home or to their offices (which I did). When being shown around the premises, I realised the inner courtyard of Facebook's contiguous buildings forms an "H". When

3. In the 1970s, Bill Gates tried to get paid for the software BASIC, and he considered it a "theft" of software when programmers downloaded it without paying for it. This didn't fare well in the programmer community. Since then, copyright and piracy issues have consistently surfaced. Being anti-copyright (and pro-piracy) is indeed linked to a discontent with authorities.

CEO Mark Zuckerberg makes announcements to the employees, he does so from the stage on this Hacker Square (see Figure 3.1).

Figure 3.1 *Hacker Square*

Source: photo by Jakob Svensson

I thus wonder what this fascination about hackers and hacking is all about. In a letter to the shareholders of Facebook, Zuckerberg (2012: 67) claims that hacking is about making the world a better place and Facebook's mission is to "rewire the way people spread and consume information" (see also Levy, 2012). Zuckerberg explains that The Hacker Way is a culture and management approach centred around five core values: focus on impact; move fast (and break things); be bold (and take risks); be open (and share information); and build social value (Zuckerberg, 2012: 70).

But, perhaps forgotten by Zuckerberg, hacking is also tightly connected to the free and open source software collaboration movements that arose from the rebellion against corporate giants like IBM and Microsoft. When Microsoft Windows was accused of hiding the workings of their systems, hackers began to couple their hacking activities with a political mission: to make information free for all. Freedom of information is thus not only connected to hippies but is also an important hacker ethic. Journalist and leading hacker expert Steven Levy (1984) states in his influential book, *Hackers: Heroes of the Computer*

Revolution, that hacking is based on access to computers; all information should be free; hackers generally mistrust authorities; hacking is based on meritocracy; hackers are judged by their hacking and not by appearance, age, or position; and finally, that hackers believe in the possibility to create art and beauty in a computer. A hacker is, following Levy, someone who sees programming as *joyful in itself*. Douglas, a Swedish programmer in his 50s whom I met in Stockholm, confirmed this. With a smile on his face, he remembered when he, in his youth, hacked some state authorities with his friends in order to get information on their love interests.

As with hippies, hackers have a philosophy of sharing, openness, and decentralisation. According to communication scholar Douglas Thomas (2002), it is by giving away code that programmers can claim to be hackers. Zuckerberg (2012: 69) actually recognised this in his letter to the shareholders:

> Hacker culture is also extremely open and meritocratic. Hackers believe that the best idea and implementation should always win – not the person who is best at lobbying for an idea or the person who manages the most people.

It is a hacker's code that matters, not their title or physical appearance. But to share something, it must be free; all information should be free, as the hacker ethic goes (see Levy, 1984). Hackers fought not only for freedom of information, but also freedom for themselves, both in terms of independence from authority and an adult world and in terms of liberation from their own bodies. The source and tool for this liberation and independence was the computer.[4]

Hacking is also about doing things because you *can* and showing off programming skills. As Levy (1984) argued, hackers do not always need a purpose – there is ample justification in the feeling of power, accomplishment, and having fun. In a recent account, media scholar Peter Nagy and his co-authors (2020) discuss hacking as the use and manipulation of systems for purposes they were not originally intended for, often for the sake of exploration, play, or experiment. To do things because it is possible and having fun at the same time connects the hacker to the prankster figure. Paul Graham (2010) even argues that if hackers would have a national day, it would be April Fools' Day. The prankster can also be traced back to the early hippies. Turner (2006) talks about a group called the Merry Pranksters, noted for a lengthy road trip they took in the summer of 1964, travelling across the US in a psychedelic painted school bus. They wanted to show Cold War America an alternative and more harmonious and fun way of life. This would resonate decades later when a hacker generation grew up with their own personal computers, using them for the purpose of pranking.

4. As much as hippies and hackers wanted information to be free and uncontrolled, it's a bit sad to see how code and predictive analytics encircles users with those like-minded. Past data traces cannot be escaped, and the individual becomes a calculable subject freely available for a computer to decipher for commercial, surveillance, or propaganda purposes, as, for example, Zuboff (2019) argues with her theory of surveillance capitalism.

The hacker figure has been linked to the cultural, social, and political history of the computer. Some argue that hackers grew out of the hippie counterculture and anti-war protests (see Thomas, 2002). Levy (1984), however, claims that the origin of hacking can be traced all the way back to the 1940s and the MIT Media Lab. The hacker figure is complex and contradictory, with different meanings over the decades. The computer programmer breaking new ground in the 1950s and 1960s has little to do with the dark and criminal connotations connected with the hackers of the 1980s and 1990s, with even sharper contrast to the consciousness-seeking free-love–practicing hippie of the 1960s and 1970s. While a hack, for Levy (1984), connoted the wild pleasure taken in involvement with the computer, the term "life hack" is today used by people to become more productive and efficient (see Nagy et al., 2020).

I return to some of these contradictions in the next chapter. Let me here focus on the hacker as a secret, ambivalent, and undecidable figure, connected to the computer underground. Hackers have often been framed as criminals of the underworld, something appropriated by hackers themselves who take on names such as Master of Deception or Legion of Doom. Mainstream curiosity and pop culture depictions signal fear for the unknown at the same time as a curiosity in technology. In William Gibson's (1984) highly influential novel *Neuromancer*, hackers are described as practical jokers and nihilistic techno-fetishists. Coming from Sweden, the character Lisbeth Salander from Stieg Larsson's novels (published 2005–2019) comes to mind. She is an outsider, extremely bright and dangerous, yet fair when she takes the law in her own hands in order to avenge her mother and childhood injustices, not least through using her computer skills.

Indeed, the hacker also incorporates certain ethics. Levy (in Fisher, 2018) coined the term "hacker ethic" to describe a common set of values hackers of any generation had. The ideal that information should circulate freely connects the hacker to hippies, as already mentioned. Information was the mystical energy circulating through the hippie communes of the back-to-the-land movement, simultaneously freeing them as individuals and binding them together in a like-minded community (see Turner, 2006). Free information is also the basis of the so-called new economy and networked mode of organising work and capital that I attend to in the next section, where I discuss tech culture's embrace of openness.

Today, many "old school" hackers have become Silicon Valley giants, betraying their own principles of freedom of information, openness, and exchange, as Thomas (2002) laments. Levy (1984) wrote about how entrepreneurship, money, and hacker stardom gained ground to the horrification of early so-called *true* hackers. He called this the age of the media hacker, not least because of cyberpunk fiction making hackers and hacking cool. Part 4 of Levy's book is titled "The Last of the True Hackers", implying there are no true hackers left.

The magic that hackers found inside the machine is now a trade secret and not free for everyone anymore. Levy (1984/2010) later concluded that there is a new generation of hackers who don't see business as an enemy but rather as a means.

Entrepreneurial & startup culture

The open office setting is conspicuous in Facebook's new building 21, where I was showed around by a proud Facebook employee. Suddenly, I walked past Mark Zuckerberg at his desk in the midst of a large open space. He looked rather mundane in his ill-fitting jeans and loose t-shirt. I was asked not to stop walking, as there was a lot of security around him (even in an open office setting).

Tech culture has indeed embraced informality. Doors are supposed to be open and status symbols removed. People are judged not by appearance but by coding skills. In *The Silicon Valley Revolution* (Tenhaven, 2017), pioneers remembered that when entering a room, they just needed to look around and find the worst-dressed person – he (seldom a she) would be the most important person to talk to. Similarly, in Gideon Kunda's (2006: 2) ethnography of an engineering division of a big tech company (one of the first of its kind and still highly relevant despite the fact that it was researched in the 1980s), he observed that "business attire seems almost theatrically out of place". According to Kunda, tech rituals have two distinct features: a decentralisation of power and an embrace of informality, openness, and individual initiative.

Today, tech culture is marinated in entrepreneurship and startup enthusiasm, with slogans such as "invent the future" and "make magic". Some (such as Kevin Kelly, in Fisher, 2018) even argue that the biggest invention of Silicon Valley is entrepreneurial and startup culture. This is particularly evident in large tech companies such as Facebook, whose work environment in Menlo Park was unlike anything I had seen up to that point, with services and food possibilities all geared towards creating an atmosphere of play, loyalty, and flat hierarchies. Kunda (2006) similarly describes managers' attempts to generate enthusiasm and a strong affiliation to the company through enforced informality, openness, and flexibility. Indeed, people working in big tech have entered into a contract that is more than economic, in that it also puts demands on how they should define themselves. This is about combining paternal care with an open, informal, yet achievement-oriented culture without "the trappings of status-conscious and rule-bound bureaucracies" (Kunda, 2006: 67), combining the playful with the semi-serious.

The organisation Kunda researched was littered with slogans about loving tech, working hard, and having fun: "It is not just work it is a celebration"; "we are a song and a dance company"; "we are like a football team"; and "we break the rules, we are aggressive". By using imagery of belonging to a family, tech

corporations underline that there should be no contradiction between personal and corporate goals. "My job is to marry them to the company", as one of the managers Kunda (2006: 23) interviewed explained when talking about his relation to his employees. This allure of offering opportunities for professional development, interesting work, relative autonomy, and fun, as well as a kind of formalised prescription of informality and semi-serious playfulness, resonated in the tech headquarters I visited during my journey into tech.

At the same time, Kunda (2006) underlines how tech companies try to enforce ideals and values through corporate ideology and workplace culture, and how managers often attempt to control a group of anti-authoritarian programmers. In the literature, an anti-authoritarian programmer is sometimes labelled as a cowboy coder, "a programmer who resists rules, prefers solitude, and likes to work on the edge" (Rosenberg, 2008: 111), who "squints his eyes, does his work, and rides into the horizon to the whistling notes of Ennio Morricone" (Chandra, 2013: 51). To tech company managers, cowboy coders are a nightmare, while they are heroes to many programmers (especially hackers). However, when I visited tech corporations in Scandinavia, Germany, the US, and India, my impression was that the strong hippie–hacker ethos of being underdogs and against authority is fading away. Walking around e-City in Indian tech metropolis Bangalore, I was almost run over by stressed-out tech workers, and in the enormous roof-top park on Facebook's building 21 in Menlo Park, only myself and some other visitors actually seemed to have the time to wind down with a frozen yogurt in the shady parlour overlooking San Francisco Bay.

There was a difference regarding this in my interview material, between those working freelance or in small startup companies and those working in big tech. Those in big tech had to embrace entrepreneurial slogans of being great and changing the world for the better. For example, when asking about heroes or role models, I was surprised by how many of the big tech employees mentioned their CEOs: "I really admire Mark"; "Satya is really inspirational"; or "Larry and Sergei were true pioneers". This is clearly an example of having to buy into a company ideology when working for big tech, but this is also about hierarchies; in big tech, there is an expectation that junior programmers listen and learn from more senior staff, as Sören, a bit higher up in the company hierarchy, told me when I met him in Microsoft Copenhagen's lunch restaurant. This is in stark contrast to the early hippie pioneers, who exclaimed "be aware of leaders, heroes and organizers" (Turner, 2006: 36).

Among the freelance programmers I interviewed, there was still a general opposition towards authority. Most of them seemed genuinely intrigued by my question about their role models and heroes. They didn't look up to anyone and had to think really hard when I tried to push them on this. Thiago, a handsome programmer in his 30s in Sao Paolo, was particularly fascinated by my interest in his heroes and role models:

> Why should I have any hero? I mean, there is this guy who contributed with some really smart code in an earlier project I worked on. But his last piece of software… I'm sorry to say, was piece of shit.

Programmers are never as good as their last lines of code, it seems. However, this doesn't mean that freelance programmers are immune to entrepreneur and startup hallelujah. Some of the freelance programmers I met had started their own business or were in the process of doing so. For example, Benjamin in Tel Aviv, whom I interviewed over Skype, described how he was building a "revolutionary tool" to manage employee expenses:

> When I was working as a programmer, I noticed that I and my colleagues were wasting a lot of time doing expense reports, and I thought, this is super wasteful of resources. Because those people are really well paid, and having them do three hours of mindless and bureaucratic work is a total waste of time and money and mind. So, I started to investigate and came up with an idea to automate that process.

When talking with Ted, an app programmer in Malmö, I heard everything about the different companies he had started. He also mentioned the "Google Way" as a technique to work with the motivation of his employees. Employees are supposed to be so involved in the business that motivation is not a problem. Inspired by Bernard Girard's (2009) book, *The Google Way: How One Company is Revolutionizing Management as We Know it*, Ted explained that in his company, everyone owned almost the same number of shares, and there was no fixed number of working hours or vacation days. It was the responsibility of the individual employee to make these decisions.

Utopian and libertarian ideals of independence and freedom can be found in both hippie and hacker cultures, but there are more traces of these in entrepreneurship and startup culture. The embrace of the weird and celebration of free thinking are also important narratives in entrepreneurship. A common thread here is a modern belief in being at the forefront and believing that programmers are changing the world for the better.

Considering this, it is hard to imagine that it was the military that hosted the early research that led up to computers and the Internet.[5] How is this possible? While visiting the Computer History Museum in Mountain View in Silicon Valley, I learned that the quest to decipher German plans in World War II seems to have been a strong motivation for developing the computer, and also for investing military funding in tech research. The story of Alan Turing, who, together with his partner Joan Clarke at Bletchley Park, broke the Nazi's encryption device Enigma, is well-known by now. According to Turner (2006),

5. We shouldn't forget (as we often do) about postal systems, the telegraph, the telephone, radio, and television, and other information technologies when we talk about the history of the Internet (see Morozov, 2013: 35).

it was in the research labs of the Cold War military-industrial complex that collaboration *across disciplines* was set as the norm for research and innovation. Brought together by a joint enemy and a tricky problem, these labs encouraged independence of mind, and researchers respected each other's expertise. In this sense, the early military research centres were quite entrepreneurial in terms of ingenuity and multidisciplinarity, where innovators, programmers, funders, and administrators were assembled, connected, and networked in order to see their projects through, much like a startup incubator. Similarly, the whole rationale behind Wiener's (1948) theory of Cybernetics was a call for interdisciplinarity, bringing together a team of specialists in different fields, joined with a desire to understand the region of control and communication as a whole.

This multidisciplinarity also resonated in my interviews. Lasse, a programmer I met in Stockholm, comes from a theatre background and has also worked as an in-house programmer at the Daily. He compared the two workplaces by underlining them both as interdisciplinary and interfunctional, with people in different roles and different areas of expertise working together. Much like a theatre – with directors, choreographers, actors, dancers, musicians, scenographers, and so on – the Daily has programmers, developers, editors, journalists, UX designers, data analysts, and ad and subscription sellers, all coming together and working for a common goal, as well as respecting each other's expertise. As hacker and artist Ray McClure phrased it when talking about Silicon Valley, "the sharing of ideas and merging of alternative cultures and alternative lifestyles, people's ability to express themselves […], I think that's the real heart of the creativity" (in Fisher, 2018: 8). Respecting each other's differences and expertise and being able to work together towards a common goal is also at the heart of entrepreneurship.

Middle-class, masculine, young, & suburban culture

The entrepreneur and the hacker are both situated in largely male Anglo-American networks. Startup culture's embrace of taking risks and believing in oneself can be connoted as masculine: "I am *the* White middle-class man", as Lasse emphasised, completely aware of the biases in the composition of the programming workforce. This leads to the next aspect of tech culture, that it is perceived of as a young, masculine, and also largely a middle-class culture.

Although not everyone today has equal access to computing technology, access in the 1970s was even more exclusive. It wasn't until the 1980s that computers were developed for household use. Early computer tinkering took place at university computer labs, only accessible to those admitted to universities having such labs. In the 1980s, when more and more people had their own PC, the largely male university culture transformed into a suburban youth culture.

Computers then became tools not only for computing, but also for communicating, a result of computers having been made personal (according to inventor Chuck Thacker, interviewed in Fisher, 2018). In this sense, it can be argued that early hacking was an activity undertaken by the educated middle class. Turner (2006) talks about a group of predominantly White men, considering themselves as part of a creative independent elite, putting the world back into balance (i.e., combatting bureaucracy) with the help of technology. This sense of self-esteem can indeed be connected to the middle class.

The exclusivity of access to university computer labs in the 1960s and 1970s also undergirded popular depictions of hackers as nerds and computer geniuses. And the computer labs were really "try-out spaces" and informal playgrounds, combining the social with the technological. Science historian Nathan Ensmenger (2015) talks about a sheltered but unsupervised environment making it possible for professional codes to be invented and developed. As one of the first to talk about the masculinisation of the culture, he describes a frat-boy culture, informed by "friendly play, rough hostility and affection through mayhem pranks and emotional aggression" (Ensmenger, 2015: 61). This idea of tech as a playground is also underlined by Thomas (2002), who describes early computer labs as spaces where young men built toys for each other. The university labs harbouring the burgeoning hacker culture were places inextricably linked to adolescent masculinity. Ensmenger (2015) talks about early programming as "brogramming".[6]

During my journey into tech culture, I have actually found programmers to be quite willing to learn as well as help others. Still, it cannot be denied that this is a masculine culture, populated by what Wachter-Boettcher (2017) argues are White guys having been told they are the best and who wholeheartedly embrace the idea that they are truly smarter than the rest of us and deserve to make choices on our behalf.[7] She calls this techno paternalism.

Tech culture has not always been this way, as Ensmenger (2015) underlines. In the beginning, coding was a mechanical, low-skilled, and low-wage job that could be conducted from home; hence, it was perfect for women, and housewives in particular. As computer pioneer and admiral Grace Hopper stated in a *Cosmopolitan* magazine article in 1976, programming is "just like planning a dinner" (cited in Chandra, 2013: 56). The earliest programmers were all women, the so-called ENIAC girls. As one of Emily Chang's (2018: 224) interviewees phrased it, "tech should be a really great job for women with families, you just need a computer, you don't always have to be in the office". In fact, the original hacker is often thought to have been a woman; Ada Lovelace is acknowledged

6. Indeed, staging contests such as the Intergalactic Spacewar Olympics has an allure of boyish competitiveness to it, a geeky arena of competition (see Turner, 2006; Chandra, 2013).
7. Worth considering, Chandra (2013: 76) argues that the gender imbalance in tech, "the particular machismo that idealizes un-socialized, high-school-outcast geekery and bro-aggression", is specific to the West.

for having programmed the first algorithm meant for a machine (see Chandra, 2013; Steiner, 2012). According to Chang (2018), women accounted for 40 per cent of computer science degrees in 1984. By 2017, this number had sunk by almost half (to 22%). In other words, programming wasn't born masculine. When demand for software increased, salaries also rose, with the consequence of men starting to populate the profession. Programming was portrayed as incredibly complex and was reframed as masculine, reformulating the practice of coding as active, creative, and unpredictable (see Ensmenger, 2012).

In addition to being middle class and masculine, tech culture is also extremely young. I saw hardly any old people walking around tech headquarters in Silicon Valley and Bangalore. My interviewees seemed unaware of this partiality towards youth when discussing biases in tech. That there is a need for more women, people of colour, and different sexualities seems, to the contrary, to have been embraced as accepted truths by the industry. When I visted Sören at Microsoft in Copenhagen, I noticed how the company proudly broadcast support of the local pride parade and had oversized signs announcing that their office is accessible for people with disabilities. At a tech conference in Berlin, there was a presentation by a tech entrepreneur and disability rights activist, Jamie Szymkowiak, who talked about better inclusion of people with disabilities in tech. In the presentation, he argued that after gender, ethnicity, and sexuality, now is the time to cater to coworkers with disabilities. Age wasn't mentioned in his talk.

I asked my interviewees if they had any older colleagues: "it depends on how you define old. We have people in their 40s in my team", as Sam, a young Chinese programmer who left Beijing to seek a better life in Silicon Valley, phrased it. Benjamin in Tel Aviv revealed that he, with his 38 years, was the oldest in his team. In Silicon Valley, I even heard rumours of an unspoken rule that if programmers are over 35 years old, they can't get hired (for an in-depth discussion on ageism in tech, see Rosales & Svensson, 2021).

It also seems that a general lack of work–life balance excludes "older" (i.e., middle-aged) programmers in general, and women in particular. In my interviews, it was striking how it was parenthood that changed programmers' attitudes towards their work. This was the case for Adam, Anders, Andri, Bart, Douglas, Hans, Niklas, Pelle, Sören, Ted, and Viktor, all of whom underlined programming as a kind of lifestyle work that became untenable when starting a family. For some of them, such as Bart and Adam, the decision to go freelance was spurred by the wish to more freely decide when (and how) to work.

Tech culture's fascination for the young is prevalent in tech historian Adam Fisher's (2018) interesting interview book about the history of Silicon Valley, "as told by the hackers, founders, and freaks who made it boom". Already in the preface, it is clearly stated that Silicon Valley is a youth culture. There are plenty of depictions of the 20-something-Silicon-Valley-CEO and how this at

times created awkward situations. The entire chapter on Atari, the legendary videogame company, is a tale of a bunch of kids having fun. There are many stories about young people dedicating their whole lives to their companies and practically living in the office. "It didn't look like a business whatsoever – it looked like a bunch of kids in their mid-twenties, you know screwing around", as Google's executive chef Charlie Ayers phrases it, reflecting on the early days of Google (in Fisher, 2018: 279). According to Heather Cairns, first employed as an assistant to Larry Page and Sergei Brin, "everyone was 20 something except for me who was ancient at thirty-five" (in Fisher, 2018: 281). When looking back at the time when he joined the company, Napster programmer Ali Aydar exclaims that he was "the older guy", even though he was only 23 (in Fisher, 2018: 283).[8] Hence, there seems to be a belief that "young people are just smarter", as Mark Zuckerberg bluntly puts it (in Fisher, 2018: 362).

The general anti-authoritarianism in tech culture can be connected to this youth orientation. The hackers of the 1980s showed a general dissatisfaction with the world of adults and expressed teenage angst, not seldom rooted in science fiction dystopic visions such as Gibson's novel *Neuromancer* (1984). Gibson also gave rise to ideas of console cowboys: data jockeys who could manipulate the system towards their own gains. This was about young people being comfortable with technology that intimidated their elders. Hacking can thus be conceived of as a space in which youth (particularly boys) could demonstrate mastery and autonomy and hence challenge parental and societal authority. Thomas (2002) therefore identifies hacking as a subculture and connects it to youth culture with its connotations of disruption and teenage rebellion.

Not seldom was young programmers' mastery of technology used for pranks at the expense of the adult world. Hacker culture gave rise to imaginations and fear of male teenagers trying to make their parents uncomfortable. Thomas (2002) thus understands pranks as guerrilla warfare: hackers staged against the adult world, while at the same time also showing that they are in control of the machine. This is described by Fisher (2018) as nerd humour, which today has become mainstream through television series such as *The Big Bang Theory* (Lorre et al., 2007–2019) and not least *Silicon Valley* (Judge et al., 2014–2019). Fisher has countless examples of this nerd humour. For example, when programmers understood the sexual possibilities of VR (virtual reality), they started using the term teledildonics. Burell Smith, described by Fisher (2018) as the "hardware genius" at Macintosh, put a sign exclaiming "Danger! Contagious Algorithm Research Area" outside one of their labs.

Tech is also a suburban culture. During my visit to Silicon Valley, I found myself driving through endless streets of one-story houses in suburban Menlo Park, Cupertino, and Mountain View on the hunt to spot some of the garages

8. Napster is a prime example, as it was created by Shawn Fanning when he was only in his early teenage years.

where it all began. Steve Jobs's garage looked rather mundane, while the Hewlett-Packard (HP) garage was turned into a small museum, with a placard outside stating that this is not only where HP was founded, but it is also the birthplace of Silicon Valley, as such (see Figure 3.2). The legendary Homebrew Computer Club (started in 1975) also took place in a garage: Gordon French's garage.[9] There is even a myth that Google started in a garage (see Fisher, 2018).[10] Wachter-Boettcher (2017) thus has a point when she talks about programmers as garage tinkerers. The car with its garage next to a one-story house in an endless lane of similar one-story houses is indeed a symbol of middle-class suburbia.

Figure 3.2 *The HP garage, the "birthplace of Silicon Valley"*

Source: photo by Jakob Svensson

9. In a true anti-authoritarian and hacker spirit, this was a computer hobbyist group which embraced an ethos of sharing, peer-to peer collaboration, and information technology as something built around community. Here it was especially the MITS Altair 8800 build-it-yourself microcomputer that attracted attention, starting the buzz around microcomputing and eventually leading to personal computing. The first Apple computer was really an extension of this terminal, according to Jobs (see Fisher, 2018: 64).
10. Indeed, Susan Wojcicki rented her garage to Page and Brin in the early days (see Chang 2018), but it is questionable whether it was here where it really started.

It is of interest to shortly attend to the differences between Westerners and non-Westerners in my interview material here. Spending some time in India to conduct interviews, and visiting the IT hub Bangalore, I observed that programming there was much more about a conscious career choice in the quest for a pleasant life. A common denominator was how programmers had followed the advice of their parents or in the footsteps of older siblings when choosing to study computer programming. Nilesh, a young programmer I met at a bustling Thali lunch restaurant, put it quite bluntly when answering why he had chosen tech: "My parents thought it would be a good career choice". Similarly, Vikas, an up-and-coming Indian programmer, told me that he saw his brother doing really well, with a pleasant life, so he followed in his footsteps. As Indian programmer Vikram Chandra (2013) also argues in his book, Indian techies are very humble and generally choose their careers based on their families' expectations. For Nilesh, Vikas, and also for Chinese programmer Sam in Silicon Valley, the entrepreneurial vocation to change the future was less pronounced than the motivation to lead a pleasant life and the possibility of living and working in Europe or the US. While not necessarily following their parents, they hadn't really challenged them either.

In the next chapter, I dig deeper into contradictions in the culture. To conclude here, it is important to underline that tech culture is not a monoculture; it has been influenced from many different perspectives, angles, and subcultures.

▼▼▼

Programmers act in and through a culture that has a history of various influences over the years, as I have shown in this chapter. Turner (2006), for example, argues that tech culture (or cyberculture, as he labels it) brings together two legacies: the research culture of the Cold War military-industrial complex and the American counterculture. Indeed, the Internet is an outcome of military experiments conducted under the Pentagon's Advanced Research Projects Agency Network (ARPANET) programme. ARPA was created to support high-risk research. In this way the Pentagon functioned as a contemporary venture capitalist for startups, with the difference being that ARPA was geared towards solving war-related problems. The first computers were war machines, and Silicon Valley was initially run on US Department of Defense contracts, and it is thus no coincidence that one of the first successful interactive graphic computer games was a wargame, *Spacewar!* (Russell, 1962). Initially, the idea of computer networks was to share valuable resources and compile processing power from many research institutions' computers when having to complete large computations. This was indeed a multidisciplinary programme. Hence, from the beginning, digital tech culture has drawn from many sources, promiscuously mingling academic theories such as Cybernetics with psychedelic drugs

and Zen Buddhist ideas. This is where the technological and intellectual output of industry and high science met with Eastern religions, LSD mysticism, and the communal social theory of the back-to-the-land movement of the 1960s. But this promiscuous mingling and seemingly unbelievable intersections between, for example, left-leaning hippies and the military, brings contradictions to the fore. I have mentioned a few in this chapter. While guided by the history of tech and its many roots, it is not always easy to navigate the tensions and contradictions that tech culture's many influences bring to the fore. Hence, in the next chapter I attend in more detail to contradictions and how programmers navigate them.

Chapter 4

Navigating a culture of contradictions

In this chapter, I attend to the contradictions in tech culture. As argued in the previous chapter, tech culture is not a monoculture; it has many influences. An honest attempt to understand tech culture and its people should zoom in on the complexities and contradictions rather than reiterate black-and-white narratives of tech as either run by purely capitalist money-making corporations, or as inherently good and changing the world to a better and more connected place. Furthermore, as programmers act in and through a culture, it is particularly interesting to explore how they navigate the most apparent contradictions – seemingly tensions – that tech culture's many roots and influences have brought to the fore. It is also in these contradictions that a culture becomes accentuated, when things must be negotiated and made sense of.

The most apparent contradiction emerging from the previous chapter is how tech's hippie roots were accommodated, first in the research labs of the US military-industrial complex in the 1950s, and then how their narrative of free information was embraced within a cheerful startup culture. After discussing this, I outline a contradiction between notions of programmers as prankish and boyish versus notions of programmers as idealistic and driven by ethical motivations. This leads me to the third contradiction, namely the preference of keeping to oneself versus seeking attention or affirmation for programming achievements. I conclude by discussing a contradiction between believing in oneself as a programmer versus being aware of limits and asking for help.

Progressive hippies vs. libertarian entrepreneurs

From the outside, the merger of the countercultural with startup enthusiasm and entrepreneurial self-confidence seems unlikely. This was apparent in the "Cannabusiness" track of SXSW2019; indeed, there was a whole conference track devoted to the business of marijuana. Coming from Sweden, with its long history of zero-tolerance for all types of narcotics, this came across as a bit unusual. One of the presentations discussed how the cannabis industry is "paving the way for traditional markets to adopt new and innovative technologies and processes". Programme highlights included sessions like, "Can We Heal Ourselves from the War on Drugs?", "The Future of Hemp

and Health", and "Why Cannatech will Disrupt Traditional Tech". The programme stated:

> We at SXSW feel that the cannabis industry is uniquely positioned to evolve into a positive force that will resonate throughout the globe. Cannabis represents a huge opportunity for entrepreneurs and innovators who are looking to establish themselves in an industry that is just beginning to mature.

At the SXSW2019 conference, I observed several instances of a capitalist–progressive contradiction. On the one hand, I witnessed outbursts of pro-profit market hallelujah, and on the other, the person attracting the biggest crowd was newly elected left-leaning Democrat Alexandria Ocasio-Cortez. I wrote in my research diary that this seems schizophrenic. Are they trying to cover everything at SXSW? Maybe this is what makes tech culture so special, that it attracts both mainstream capital and progressive politicians at the same time?

The fusion of hippie bohemianism with the high-tech industries of Silicon Valley – marrying business with a hippie ethos of the left – is sometimes referred to in terms of a Californian Ideology. Media scholars Richard Barbook and Andy Cameron (1996: 45) argue that the whole idea of the Californian Ideology is to combine "the freewheeling spirit of the hippies and the entrepreneurial zeal of the yuppies". Indeed, California has a sense of experimentation to it, an openness to new possibilities. According to Steve Jobs, in the early days of Silicon Valley, "you could get LSD fresh made from Stanford and you could sleep at the beach at night with your girlfriend" (in Fisher, 2018: 60). Californian Ideology embraces the visions of both the left and the right and is thus a hybrid between progressive values and "capitalism of innovation", wired together in the belief that technology can induce change. Indeed, Chandra (2013: 64) uses the label hippie capitalists. Free-market ideals and a general anti-statism are normally attributed to the right wing in politics. However, anti-statism also resonates in the counterculture of the 1960s, with hippies reacting against the oppressive authorities of the state.

This focus on the individual combined with a general anti-authoritarianism was to be picked up by right-wing politicians when they came to embrace the notion of cyberspace. According to Fred Turner (2006), ideas of a new economy started to circulate in the 1990s, when threads of technology consciousness, decentralisation, and personalisation attracted the attention of Republican politicians in the US. In the new economy, employees were conceived of as entrepreneurs moving in and out of collaborative teams, not least aided by computer technology. The hippie aversion to bureaucracy as a mechanistic and destructive force was coupled with the general anti-statism of the right wing, and the urge to render bureaucracies obsolete united hippies and Republicans. Freed from the institutions that structured privilege in the material world, it was the individual that should join society. Hippies underlined a more holistic

individuality, not seldom informed by systems theory and Cybernetics, in which biological systems maintain order by means of evolutionary forces. In other words, the individual had to be liberated from oppressive authoritarian forces such as American state bureaucracy. This thinking was attractive to Republicans. Although not hippies (or hackers, for that matter), proponents of a so-called new economy shared an affection for empowering technologically enabled elites, building new businesses, valuing decentralisation and personalisation, and rejecting traditional forms of governance.

Adam Fisher (2018) presents several entertaining illustrations from when hippies and corporate America began to intermingle. Some of his interviewees questioned whether the future could really be invented by a hippie "who didn't even own a car" (Fisher, 2018: xiv). The friction between mainstream business people and hippie entrepreneurs is particularly apparent in the story of the gaming company Atari. Prior to Atari, Silicon Valley was filled with men in suits, but with the advent of Atari, "Silicon Valley became hippies in jeans smoking weed" (Fisher, 2018: 60). Apparently, business executives labelled these hippie entrepreneurs "the great unwashed". One of these strange encounters is when an investor came to Atari dressed in a business suit and tie and met with Nolan Bushnell, one of the founders, who was wearing a t-shirt with the text "I love to fuck" written all over it. Mark Zuckerberg (in Fisher, 2018: 355) states:

> Most businesses aren't like a bunch of kids living in a house, doing whatever they want, not waking up at normal time, not going into an office, hiring people by, like bringing them to the house and letting them chill with you for a while, party with you and smoke with you.

Reading Fisher's concluding chapter, it seems, though, that entrepreneurship culture has washed off some of the aspects of hippie behaviour in tech. Gideon Kunda (2006) argues that values in strong corporate cultures tend to permeate culture at large. Maybe this is why entrepreneurship values are becoming increasingly mainstream in contemporary tech culture today.

Burning Man

The idealistic hippie has not completely exited from tech culture, as exemplified by the continued embrace of the Burning Man festival, an annual event that started in San Francisco but is now held in the Nevada Dessert. Mark, an American programmer I met in San Francisco and later interviewed over Skype, talked fondly about this as an event at which more holistic and collaborative ideals are celebrated. The event is based on participatory ideals and explores various forms of artistic self-expression created to be enjoyed by all participants. According to Turner (2009), Burning Man's bohemian ethos supports new forms of production emerging in Silicon Valley. Practices such as building

sociotechnical commons, participation in project-based artistic labour, and fusion of social and professional interaction drive the growth of Google and other Silicon Valley businesses. That Burning Man is important for tech is also apparent in Fisher's (2018) chapter on Google. The first Google Doodle was, for example, made out of the Burning Man logo, and it is described how some of Google's more outrageous ideas (such as "to rule the world") were born at the 1998 Burning Man festival. In the early days, Google is even said to have used a "has visited Burning Man" criterion for employing key staff, such as the CEO Eric Schmidt. Though not a tech event, it still appeals to programmers, because it is "a complete alternative reality, a different world" (according to Michael Mikel, aka Danger Ranger, in Fisher, 2018: 245).

Underlying both ideas of a new economy and hippie holistic individuality is a belief in a network mode of organisation. The network became the principle of a new society, with the Internet as a model of a decentralised and de-governmentalised society (see Turner, 2006). Information technologies were supposed to empower the individual, enhance personal freedom, and reduce the power of the nation state. Power structures were supposed to be replaced by interactions of autonomous individuals and their software (see Barbook & Cameron, 1996).

This thinking also brought along ideas of mobile, flexible, and decentralised ways of working, ideas attractive to companies with employee liability. Right-wing politicians thus embraced digital technology (with the network as its prime mode of organisation) and coupled it with ideas of liberating individual entrepreneurs: a world where man creates his own destiny. Hippies also embraced the idea of a flattened and network type of organisation in their quest for a global society. For the hippie pioneers, though, computer technology was a tool for collective transformation, to augment the human mind in the service of humanity, with the goal of building alternative communities (see Turner, 2006). Hence, while computers were tools for decentralisation and flexibility for the right wing, for hippies, they were tools to connect and transform the mind and consciousness.

Today, programmers still value freedom of information and strive to make the world a better place. Nadja, a programmer and entrepreneur in her 50s whom I interviewed in San José, lamented how the culture had moved away from open source "where it all started", where people worked for free and invested their time because "they believed in something". Now, she claimed – with a bit of sadness in her voice – all initiatives are monetised and there's a wave of people wanting to "make a buck on everything". "They talk about helping people, but now it comes with a price tag", she said. She compared this with the early programmers (whom she referred to as "true believers"), who went to coun-

tries, such as her native Lithuania, to help people set up networks, using their own money to do it: "At this time there were no fundraisers, no donations, no venture capital", she said, and continued to underline how much she missed these more idealistic times.

Hippies and entrepreneurs are also libertarians, all underlining the freedom of information and individuals, which might explain why it seems easier to be gay than conservative in tech. In the Skype interview with Benjamin in Tel Aviv, I asked him about being homosexual in a relatively homogenous group of programmers. Benjamin explained that it didn't cause him any problems, and that it was worse for the religious person in the team:

> No problem, not in Israel. My previous work was homogeneous in terms of people working there. There were quite a lot of women coders, relatively; it wasn't half, but close to 40 per cent women, which is relatively a lot. But they were all Jewish, and even from the political map, they were all in the same mindset, like the middle-left of the map – relatively progressive. So, me being gay was a non-issue. But we had one programmer who was a religious person. He was the only one and he was the outsider, and he was the one challenging the group socially. Being gay, in comparison, was easy.

It seems it is this ability to work together and listen to each other – respecting each other's expertise – that is needed to navigate the tension between progressive hippies and libertarian entrepreneurs. The programmers at the Daily in Stockholm, for example, staged an in-house stand-up comedy event once a week. Viktor, one of the programmers, underlined the general good mood this created and coupled it with the purpose of working together to achieve a quality product: "You all need to come together and work for a common goal, a common product. And to work together, you need to have a little bit of fun". Having a common goal and cherishing the individual as one node in a flexible network – respecting each other's freedom and expertise – seems important to navigating between entrepreneur and hippie ideals.

Pranking dudes vs. techno-missionaries

The playful prankster, sometimes represented in the darker figure of the hacker, does things just for the sake of it, because they *can* and because it is fun. This can be summarised with the notion of "lulz" (deviated from LOL, meaning laugh out loud). To do something "for the lulz" was used to justify ridiculous, pointless, and occasionally gratuitous behaviour – behaviour that sometimes stands in contrast to making the world a better place, as Nadja talked about. In fact, my interview material is full of so-called techno-missionaries (as Darrah labels them in an article from 2008).

When asked about their life in tech, programmers tended to highlight their more charitable work. One example was Roger, an older programmer I met at Google's headquarters in Mountain View, who reflected upon his background in Africa setting up mobile services: "It was then my real passion started to emerge… and that is, and has always been, how technology can change lives for the better". He acknowledged there were two sides of the technology coin, but he was all about finding "the positives and pushing those to the limits". Roger thus underlined how mobile technology helped people in Africa to increase productivity, helped them to learn English, and made healthcare information accessible for all (for an overview of the field of mobile communication and development, see Svensson & Wicander, 2010). According to Roger, this is was what gave him a reason "to get out of bed and fly to Nigeria once every month". Then he worked in Uganda with the Bill and Melinda Gates Foundation before ending up at Google in Silicon Valley. Throughout the different jobs, he retained the same motivation: "trying to have a positive impact with technology".

Martin, a young Dane I interviewed at the Facebook headquarters in Menlo Park, also underlined that the most rewarding thing for him was to help others. He proudly declared how he had worked with kids with autism, people with Amyotrophic Lateral Sclerosis (ALS), people with disabilities, and how, through his technological innovation, he was able to give them a better life:

> Being able to spark life in these people, being able to give XX technology to people was very rewarding, and that is why we founded our company, *to make the world a better place*. […] I always tell people, *if you start a company, you should do it for noble reasons*; if you do it for money, you will not succeed. When we started building XX technology for 99 USD instead of 25,000 […] suddenly we started getting mail from Africa. We got mail about people who could communicate again – "I can now communicate with my father" – *that was the most rewarding thing*. [emphasis added]

Martin claimed that his company "made the world a better place" and that this is the achievement he is most proud of today. He had similar praise for Facebook, where he now works after it bought the company he worked for previously. He described it as "a safe haven for people of different minorities", mentioning how people of colour, transsexuals, and also programmers from Pakistan and India work together doing "wonderful stuff". According to Martin, who programmers are and where they come from is not what is important, because "we worry about making the world a better place through our products and services". Sam, the young Chinese programmer in Silicon Valley, also mentioned as his role models people "who make something happen, something influential that can change people lives for the better". This made me think about Google's famous code of conduct, "don't be evil". To do no evil is about "doing the

right thing", and all work at Google "should be measured against the highest possible standards of ethical business conduct" (Conger, 2018: para. 1–2).[1]

In my interview material, it was mostly the freelance programmers who could afford to make ethical choices regarding who and what they chose to work with. A majority of them could pick and choose among projects (due to a general lack of programmers, at least in Sweden). Niklas, whom I met in the small Swedish town Norrköping, told me he mostly worked with climate change projects. Pelle in Stockholm underlined how he would never work with gambling, alcohol, or tobacco companies. Also, Python, a middle-aged freelance programmer I interviewed in Berlin, proclaimed he would never work for a nuclear power plant or the military. At the Daily, a majority of the programmers I met had chosen to work at the newspaper because they believed in the Daily and generally valued traditional journalism and were sympathetic towards its democratic mission. Viktor expanded on this:

> I am quite interested in societal matters you know, and it is fun to work with a product you care about, a luxury to think that what you do is good… and to feel the pulse from the newsroom and journalism.

Interest in social issues was also emphasised in an interview with Anna,[2] an older non-binary programmer I interviewed in Oakland, who underlined that "engineers are much more egalitarian than you would think they are". Anna herself decided to work in tech because of her interest in gender issues. She emphasised that she had never had any issues with the programmers themselves; instead, it was the *management* who posed the biggest problems to her endeavours of trying to make tech more equal.

Tech is indeed a conscious culture, and the companies embrace environmentally friendly practices such as cycling and recycling. Google provides their employees colourful bikes for free, and at Facebook, one of the many free services for their employees is a bike repair shop, attending to employees' bikes while they are at work. Even in plastic-flooded India, the recycling initiatives in the IT hub Bangalore were clearly visible, and in Scandinavia, tech companies happily broadcasted their environmentally friendly strategies as well as their support of local pride parades.

The ethical and progressive side of tech was also visible all over the tech conferences I attended. At the Öredev2018 conference in Malmö, toilets were gender neutral, and eco-friendly vegan ice cream was being served during the breaks. The conference bag was made of renewable material, something which was clearly announced on it: "they say I was trash, now I'm fashionable".

1. Apparently, Google dropped "do no evil" from their code of conduct in 2018 (see Conger, 2018).
2. Though presenting herself as non-binary, she expressed a preference for female pronouns and name, which I have adopted here.

The conference water bottle had "We Care" printed in bold letters under an illustration of Earth. Conference attendants were further informed that the bottle was for tap water only, made out of BPA-free plastic, and dishwasher safe. In the foyer, I also learned that the conference was sponsoring a tree-planting nongovernmental organisation in South Sudan: "save the planet one tree at a time". Attendees could join yoga classes in the morning, and conference goers were reminded that bio-hacking (the theme of the conference) is ethical. The narratives from Öredev were reiterated at the React Day conferences I attended in Berlin in 2018 and 2019, whose tagline was, "Build code, not walls".

Indeed, everything at tech conferences seems sustainable and conscious. The exception is perhaps a faiblesse for sugared carbonated drinks, with several attendees seeking out Coke despite the efforts at Öredev2018 to promote tap water. A caricature of an unhealthy programmer sitting long hours in front of a computer in a dark room sipping on soda in order to get some energy to continue working popped up in my head. This also resonates in cyberpunk novels I've read: programmers as unhealthy and asocial libertarians, lone individuals fighting for a virtual existence of information in which their unattractive physical bodies won't limit them. Indeed, there is something that doesn't fit with the image of the conscious techno-missionary. The general embrace of the transhuman, androgynous, and sometimes transsexual bio-hacker chafes against the nerdy programmer in his checked flannel shirts, oblivious of his physical appearance.

I was also somewhat surprised at how important it seemed for many of the presenters at both the Öredev and React Day conferences to be "just ordinary". It was striking how often the speakers mentioned their families, with constant reminders of how their partners tease them, or how they tease their partners, and with quite a few pictures of spouses and kids included in the presentations. One even brought her husband along and proudly presented him to the audience at the beginning of her talk. Another asked the audience to participate in recording a happy birthday message for his son. This was followed by another asking us to help him record a birthday message to his girlfriend. Yet another asked us to scream in the background so he could record it and send to his daughter, who didn't believe he was speaking in front of 800 people. I noted to myself that this seems to be something important to showcase: that they actually have a family, that they are just like everyone else, that they fit in mainstream heterosexual society.

Emily Chang (2018) uses the terms "dude" and "bro" to describe the figure who places emphasis on mainstream heterosexuality. Fisher's (2018: 95) interview book is full of examples of this: men working hard but also playing hard, with a very peculiar sense of humour, talking about a "limp dick society" and pulling pranks like delivering exploding pizzas and setting pianos on fire. Apparently, Mark Zuckerberg's first business card had "I'm CEO…

bitch" written on it, and in the first Facebook office, they had lesbian love scenes in the restrooms. People wore pyjamas to the office, and at four o'clock every afternoon they would have a meeting about "how to get fucked" that night (see chapter 25 in Fisher, 2018). As Chang (2018) underlines, this is not a particularly pleasant environment for female programmers (and probably many others), and this also puts into question whether tech is actually based on meritocracy or whether a bro-type mentality is needed to succeed in Silicon Valley. Chang (2018) describes how drug-heavy sex parties are justified with references to progressiveness and open-mindedness, and how these parties were like Christmas for lonely nerds who all of a sudden had been catapulted into fame while not really knowing how to behave around women.

The contradiction between what can be labelled a pranking trickster and a techno-missionary is nicely captured in media scholar Sylvain Firer-Blaess's (2016) thesis about the collective identity of the hacker collective Anonymous: when Anonymous went from poking fun at authorities and pulling pranks to being more focused on activism and political hacks, Firer-Blaess outlines a schism within Anonymous between so-called lulzfags and moralfags. He thus argues that these two figures – the trickster and the do-gooder – form the basis of a multipolar system out of which Anonymous acted, but also how these two figures created tensions within the collective itself.

The story of Anonymous shifting from pranking to activism resonates in other parts of tech culture. Steven Levy (1984) describes how hackers started to become interested in activism when the world of hacking gradually moved from Tech Square in Boston to northern California in the 1970s. Today, programmers are more often motivated by a higher purpose; they want to help people and do good, and they believe that indeed they *can* do good. Anthropologist Charles Darrah (2008), in his article with the revealing title "Techno-Missionaries Doing Good at the Center", underlines Silicon Valley as a centre for a progressive force of global change – "the Mecca of the technological age", as one of his interviewees phrased it.

I've attended to the contradiction between a heterosexual dude cracking jokes at the expense of others "for the lulz" and the anonymous activist with powers to overthrow corporations and politicians. These figures have given way to the less nihilistic and more optimistic entrepreneur. But what unites the prankster and the techno-missionary is the pleasure of exploration and breaking new ground, together with a belief in their ability to use the computer to achieve their goals. Douglas in Stockholm exemplified the maturing "do-good programmer" when recounting how, in his youth, he explored the computer with his friends to hack state authorities, and now he was trying to make the world a better place with his work. Early programmers and new communalists also imagined a new and better world, which is in sync with the sixth principle of Levy's (1984) hacker ethic: computers can change life for the better. Still, it seems

that a general prankish attitude rewarding masculine humour is prevalent in tech today, as well as a preference for working undisturbed out of the limelight.

Shy outcasts vs. attention seekers

An interesting aspect of tech culture is its embrace of working in the dark, undisturbed and out of public attention. Especially the freelance programmers I interviewed seemed to have embraced the possibility of keeping to themselves. For example, Andri, an Estonian programmer I met in Malmö, underlined his preference to be alone. Indeed, many of the programmers I met told me they were shy, and some even felt like outcasts. For example, San Francisco–based programmer Mark described how he "was kind of shy, queer and didn't understand school". He felt like an outcast and found a lot of comfort in going to the computer lab after school. The computer lab and the world of ones and zeroes became his refuge, as this was a space in which he felt he was in control and could detach himself from his physical – and in his opinion, un-cool – body.

Not all of my interviewees preferred to keep to themselves though, but when this was the case, they were aware that they are unusual. Bart, the Dutch programmer – who now, after a burnout and becoming a father, travels across Europe to give talks at meetups – understood that he was unusual: "I can program, I understand technology, and I can communicate about it, which is actually kind of a rare skill among software people". He explained to me that there were different kinds of programmers:

> [There are those] who just like to sit in a room with no windows in front of the computer – just do their work – and they get satisfaction from that. […] They don't want to talk to people, they have a problem looking you in the eye, and are usually a little bit autistic.

Ted in Malmö, having co-founded a company where he was now CEO, explained why he took on this position, and at the same time underlined a preference among programmers in general to work alone in the dark:

> I have a fairly unusual characteristic, which is that I like to spend time with people. So that is why I have the role I have in the company. I also believe that I am the one in the group who finds attending meetings the least boring. I actually find that the most fun part of the job – to meet people. Everyone else is more introverted than I am.

Most programmers were driven by their love of coding, "to build the best things, things you are proud of", as young Gabriela in Sao Paulo phrased it. But this also entails seeking attention for one's programming achievements. "When I have written some beautiful lines of code, I want my colleagues to

know about it", Gabriela continued. Similarly, Google's algorithm PageRank was apparently named after one of its founders, Larry Page, not the web pages it ranks. One of the most famous examples of this pride of authorship (or attribution of recognition) is the operating system Linux. Linux was first released by Finnish-American programmer Linus Torvalds in 1991. Allegedly, Torvalds preferred to call his invention Freax (a combination of "free" and "freak", with the "x" alluding to Unix), as he thought Linux was too egocentric. However, his co-worker Ari Lemmke wasn't fond of the name Freax and thus named the operating system Linux without consulting Torvalds, who, however, eventually consented to the name. Torvalds ambivalence is rather characteristic of this contradiction; indeed, a pride of authorship isn't easily combined with a general contempt for people who seek attention. As long as it is programmers' work that they are recognised for – their code, rather than their persona – it seems okay. As referred to by Levy (1984) in his outline of a hacker ethic, hacking should be based on merit, and hackers should be judged by their hacking and not by appearance, age, or position in the physical world. Fisher (2018: 215) depicts a programmer (Jamie Zawinski) as having a very peculiar sense of hairstyle (with half his head shaved), but it didn't matter because "he was a brilliant programmer". Indeed, both Levy's and Fisher's books are full of references to a supposed meritocracy among hackers and Silicon Valley pioneers.

Nonetheless, I cannot help but think that the hacker ethic of freedom of information becomes somewhat curtailed when claiming programs through naming practices, such as in the Linux example. As programmer and science fiction writer Vikram Chandra (2013: 66) underlines:

> It is within open source that programmers most fiercely pledge allegiance to the legacy of the early neckbeards. And so Linus Torvalds, the "benevolent dictator" of Linux, dismissed the makers of rival operating systems as "a bunch of masturbating monkeys".

Still, most programmers know they depend on others, that they build upon others' code: "Programmers today only program on half of the function, they don't program on all of it", as Chris Dancy explained when I met him a second time in Austin. Programming depends on information being free and distributed. Data scientist Pedro Domingos (2017: 94) writes that "in a connectionist system, the answer is 'it's stored a little bit everywhere'". Indeed, the ethos of sharing, with its roots in hippie culture, is pivotal and can be connected to the importance of a community of programmers. People who have been described as stubborn individuals or rational-minded loners (see Rosenberg, 2008) also need a community where their work can be recognised and acknowledged. This is most apparent when discussing good code. Good code is *understandable* code. Many of the programmers I interviewed underlined that other programmers should be able to understand and use their code. Hence, it needed to be open and freely available.

In this sense, even though programming seems to be a lonely profession – in that programmers prefer to keep to themselves – programming as a practice is situated in a social context in which programmers not only speak to the machine, but also to a community of other programmers. This implies that even if there is a faster, more space-efficient way to write code, it isn't preferred, because other programmers might not understand what is going on in these lines of code (if not properly named, tagged, or explained). To impress peers is important, but if programmers trim code to the fewest possible lines, in some kind of masculine competition (see Ensmenger, 2015), they might lose recognition in the community for their work.[3]

However, relationships between programmers are different than what one might think in terms of other more conventional types of professional relationships. It is close in terms of their shared love for the computer and its supposed potential, "but not imbued with the richness of a real-world relationship", according to Levy (1984: 129). John Markoff (2015) talks about loners, and Scott Rosenberg (2008) writes about a geek syndrome, referring to a preference for working alone, avoiding eye contact, and a general difficulty reading body language. Douglas Thomas (2002), writing about hacker culture, claims that hackers have been known for their love of late-night junk food and general slacker attitude. Nathan Ensmenger (2015) talks about this in terms of a computer-bum phenomenon: spending the whole night in the dark, with only the computer screen as a source of light as he gets into the flow, absorbed with a problem he seeks to solve. This is connected to what is labelled a compulsive programmer phenomenon – that is, spending long hours in dark computer labs, neglecting bodies and physical appearance – a label arguably coming from legendary computer scientist Joseph Weizenbaum:

> Bright young men of disheveled appearance, often with sunken glowing eyes, can be seen sitting at computer consoles […,] their rumpled clothes, their unwashed and unshaven faces, and their uncombed hair all testify that they are oblivious to their bodies and to the world in which they move. These are computer bums, compulsive programmers. (in Levy, 1984: 129)

This is about working together through the machine, pushed towards ideals of meritocracy, in which programming skills are privileged over titles and looks. As Levy (1984) asserts, beneath early programmers' often-unimposing exteriors, they were adventurers, visionaries, risk-takers, artists, and the ones who most clearly saw why the computer was a revolutionary tool. His description of Steve Wozniak is particularly to the point here:

> He looked like a bum. His hair fell haphazardly on his shoulders, he had the kind of beard grown more to obviate the time-consuming act of shaving than

3. There is even a language called "brainfuck" which was created as an exercise in writing the smallest possible compiler (see Chandra, 2013).

> to enhance appearance, and his clothes – jeans and sports shirts, with little variation – never seemed to fit quite right. (Levy, 1984: 249)

This might have to do with secrecy being tightly connected to hacking. Thomas (2002) even argues that hacker culture is a culture of secrecy. In a way, this is in opposition to both the pranking dude and the techno-missionary. The prankster seeks to have an effect with his pranks, that at least the jokes should be noticed. A missionary is per definition someone who seeks an audience for their message. But the secrecy of hacking isn't without its contradictions either. On a local level, the anonymous hacker deploys secrecy strategically in order to avoid law enforcement. But on a global level, the practice of secret code – hiding information from users – is to be destroyed, as evidenced by the discontent with IBM as previously discussed.

Geeks in public

When geeks are taken out of the dark and expected to have fun in public together with others, it sometimes creates tensions and awkwardness. At Öredev2018, there is an overtly positive cheerful atmosphere fuelled by a super enthusiastic moderator already at 8:30 in the morning, which strikes me as unusual. "Isn't this great!" he exclaims, and the crowd cheers back. "Raise your hand if you are a Star Trek or a Star Wars person", "turn to your neighbour and talk for two minutes". At one point he even gets us to participate in the children's game Raketen [the Rocket]: everyone is clapping, from one side to the other, and we clap faster and faster as the imagined rocket is about to launch, and then everyone jumps in the air when the launch moment finally arrives. Not everyone, including me, is comfortable participating in this collective cheer and – what seems as entrepreneurship-tainted – hallelujah. At the React Day conference in 2018, I find myself again listening to an overtly enthusiastic conference moderator who forces me to interact with my neighbour: I'm instructed to ask his name and then scream that I love him (!). The moderator delivers some jokes: "What do you get if you cross a computer with McDonalds? – A Big Mac". People laugh dutifully. "How can a computer get a cold? – You open too many windows". Some programmers seem to thrive in this cheerful environment, but what I also witness is the lone computer nerd meeting the ultrasocial and cheerful entrepreneur – and this meeting isn't always comfortable.

In the React Day 2018 conference programme, the speakers are depicted as cyborg-like cartoons, with machine parts and cords hanging from their heads. When it is time for the speaker to enter the stage, the cyborg cartoon is projected on to the big screens and a stroboscope starts in tandem with loud and energetic music. The speakers are presented as cyborg rock stars; some of them are better at taking on this role than others. For example, one software developer who engineered a program for making music beats showcases it in his talk without taking off his sunglasses during the entire presentation. He

sways along with the beats he produces live and even moves his hand to his mouth as if he's smoking weed while chilling to the cool tunes he is producing. Other speakers, however, seem less comfortable – and even a bit lost – having to start their talk about some very specific programming problem just after being presented as a cyborg rock star.

I write in my research diary that tech nerds have become cool, but perhaps the tech nerds themselves haven't fully kept pace with this transition.

The contradiction between the ultrasocial bro and the nerd who just wants to have an ordinary family life is striking. Chang (2018) traces the start of the transition from the nerd to the rock star (or bro as she labels them) with the rise of Apple and Steve Jobs. But the rock star image goes further back than Steve Jobs. Young programmers were photographed by Annie Liebovitz for an issue of *Rolling Stone* magazine already in 1972; the photos were part of a piece about the legendary computer game *Spacewar!* (see Turner, 2006). Here, programmers were compared to rock stars, technologically savvy and counter-culturally cool. The hacker was no longer necessarily only a nerd, they could also be an icon (see Turkle, 1995).

In the meetups I attended, the contradiction between keeping to oneself and striving for attention is perhaps the most visible. Who gives away information? Who is listened to? Who is considered as having something to say? There is a stratification along these lines. However, it seems important that programmers are not perceived of as *consciously* seeking the limelight; it should be more like having been pushed out there accidently. Bart, for example, described finding out that he was good at talking in front of others as a coincidence, something "that just happened". To become a respected speaker within a particular programming community was also underlined by Sören at Microsoft in Copenhagen, who emphasised how his skills in database architecture had gained him some respect and proudly stated that he was at the same level as the speakers of the conferences he attended and was sometimes invited as a speaker himself. Indeed, the conferences and meetups I participated in do underline a kind stratification between those who share information and those who seek information, and how some (while far from all) feel at ease as entertainers, or being projected as rock stars, when speaking in front of others.

Self-confidence vs. asking for help

The final contradiction I want to discuss is that between believing in oneself versus recognising one's limits and asking for help. There are some big egos in tech culture, not least in connection to the culture's entrepreneurship and

startup influences.[4] Nadja in Silicon Valley described good programming as "designing the impossible", which indeed requires some self-confidence. Similarly, Bart talked about his hero Elon Musk as someone who "just does it, he asks for forgiveness later, and he isn't afraid to fail". Elon Musk is known for being self-confident and believing in his ideas, even though it seems hard for some of them to kick off. To strive for attention implies believing you have something worth seeking attention for. Sören, for example, talked about himself as an expert going into teams to "build up expertise, and then I leave, as I'm not needed anymore". He could then move on to something else, a new field where he could become an expert, teach others, and so on. Talking with him, I had no doubt he believed strongly in himself.

At a falafel restaurant on a chilly evening in Berlin, I met Kristina, a Polish woman in her early 40s who was changing careers and had started studying programming. She connected this self-belief and confidence to tech culture's male orientation: "We women aren't trusting ourselves enough". According to Kristina, when men apply for programming jobs, they look at the eligibility criteria and if they match about 60 per cent, they apply; women, on the other hand, think they need to match all requirements in order to apply. Almost exactly the same story is told by women in Chang's (2018) book – that they feel less confident than men. Ensmenger (2015) further supports this, underlining that the importance of believing in oneself as programmer and being able to sell yourself and your skills to employers are pivotal in the masculinisation of the profession. Self-confidence, masculinity, and entrepreneurship seem to be interconnected. According to Chang (2018), it was with the rise of Steve Jobs that tech investors stopped looking for asocial awkward nerds and gravitated towards über-confident entrepreneurs.

At the same time, self-confident experts need people who are willing to ask for their help. And to ask for help implies there is a community of programmers to rely on for this help, which is connected to the value of sharing and building upon others' expertise. It is, however, noteworthy that a culture that celebrates the ethos of not giving up (grit, see Chapter 6) and the self-taught genius (see Chapter 5) still underlines the value of asking for help. To learn, programmers need someone to learn *from*. The importance of meetups underlines tech as a learning culture, where programmers are eager to listen to each other and build upon each other's code and solutions. "We want to learn and have fun", Ted in Malmö explained, and continued, "the main purpose isn't to make money, but to have fun at work, learn, and earn enough to make a living".

Given this embrace of learning and the joy of new things, it is interesting how formal education is frowned upon among the programmers I interviewed. A majority of them stated that education was not necessary to become a good

4. "It's Ok to have dreams […] and make them reality", according to Steve Wozniak (in Fisher, 2018: 1).

programmer: "You don't need any formal education at all, only interest", Lasse asserted. We talked about education when I met him in Stockholm, and Lasse clearly spelled out his scepticism towards education:

> I don't think you really need it to become a software developer. I've met so many people now the last couple of years that are self-taught. They just open a book or a website or whatever, there is so much information… and they just learn it themselves you know. The thing that you need is the ability to learn. That is the thing that you need… and curiosity… and not being afraid to fail.

It could be argued that universities represent some kind of authority. And tech culture, with its roots in hippie and anti-war movements, appreciate neither hierarchies nor authorities. The culture is full of legends about college students dropping out of school to start a business and then making millions, most famously Bill Gates and Steve Jobs (for more examples, see Dadson, 2016). Anna, the non-binary programmer in Oakland, discussed this as a narrative in the culture – programmers conceiving of themselves as not formed by university experience. Evgeny Morozov (2013) considers this tech solutionism at is purest, that the world's problems are so urgent, and the tech in our hands so great and mighty, that programmers cannot wait a couple of years to graduate.

Still, this contempt of formal education is somewhat misleading, given the proximity of both Stanford and Berkley to Silicon Valley. According to Steve Jobs, Silicon Valley was the place where rock 'n' roll really happened, because Stanford and Berkeley attracted smart people and deposited them in a clean sunny place with great food and, at times, a lot of drugs and fun, so they stayed (in Fisher, 2018). Additionally, Facebook launched at the Ivy Leagues. Perhaps these universities are more a symbol for the privileged middle-class dude (see Chapter 3) than for the importance of formal education. As Facebook programmer Scott Marlette explains, Facebook launched at Stanford and Harvard because that's where Zuckerberg's friends were (in Fisher, 2018). It's easy to agree with Chang (2018) that when these young men – who started out at some of the world's most prestigious universities – claim to be outsiders, it's a bit of a stretch. According to her, Silicon Valley is all about hiring buddies and starting companies with bros. Fisher (2018) also has plenty of examples of this. But while Fisher labels Silicon Valley pioneers as geniuses, Chang (2018) points to the importance of social ties from attending elite universities.

Downplaying formal education and highlighting the ability to learn on your own and acknowledge what you don't know seems to be important in the culture. For example, when I tried to flatter Nadja, emphasising her education and the successes she's had in Silicon Valley, she seemed obliged to underline that she was "still dumb". I was surprised to hear this, and I asked what she meant: "Programmers say the more you know, you realise you know less and less".

She thus underlined an ability, as well as a *willingness*, to learn. And to learn, programmers must realise they do not know everything and should ask for help.

The importance of asking for help was also underlined by Hans, a middle-aged German freelance programmer, now based in Malmö, who connected his increasing willingness to ask for help to having become a parent. Ted – also a parent – had even tried to implement a culture in his startup company where programmers should be quick to ask for help. Similarly, Sunil, a young Indian programmer placed by his company in Berlin, explained how he had learned from his mistakes and how he now knew when it was time to ask for help. At the same time, Sunil highlighted that he was a perfectionist and one of the best in his team. He explained that programming languages change quickly and programmers must stay on their toes. Good programmers – while still brilliant – will *always* need to learn from others.

Curiosity is an important characteristic for navigating the tension between asking for help while still believing in oneself. Kristina, the Polish student in Berlin, was clearly driven by her curiosity, leaving everything behind to embark on a new career. I asked her if it also took courage: madness, rather than courage, she answered, laughing, but continued to underline that what made her take the step to leave her former career and begin studying programming was first and foremost curiosity. Similarly, Benjamin in Tel Aviv asserted that even as a kid he was curious, and this curiosity drew him to programming. Many of my interview participants also referred to online courses and tutorials as where they would go to learn things. When it comes to being curious about new stuff and not afraid to fail, Sunil was an excellent example: he had taken up studying things from embroidery to deep learning in neural networks. I asked him if he wasn't afraid to fail or whether completely new things didn't scare him. The fear of missing out seemed scarier:

> *No*… that is what I say when I meet people and they are like, "isn't that too advanced?" And I am like, look at how I started. It is *not* difficult. Not even one month and you will pick it up. Let's give it a try, let's see what happens. […] I'm afraid, but I'm more afraid of missing out… it is like *I* should have this experience. And obviously with everything you do, you also learn. [emphasis original]

Curiosity and confidence – not being afraid to explore new fields and ask for help in the process – do not seem to be in opposition in tech culture. This can be connected to the early hackers and their curiosity for exploring and breaking new ground. Curiosity has always been connected to hacking and still plays an important role in tech culture. As a hacker in Thomas's (2002: 97) book phrases it, "we explore, we seek after knowledge, my crime is that of curiosity". Levy (1984: 17) similarly describes hackers as motivated by exploration and as science-mad people "whose curiosity burned like hunger" (see also Graham,

2010). In the *Frontline* documentary episode "Amazon Empire: The Rise and Reign of Jeff Bezos" (Jacoby et al., 2020), viewers are told about how Bezos was trying to brand Amazon as less predatory and more loving through juxtaposing the figure of a conqueror with an explorer. Championing curiosity, the explorer was the figure Bezos wished to connote Amazon with.

In conclusion, programmers seem to conduct a balancing act between believing in themselves while at the same time downplaying their ambitions, being accidently successful and emphasising their dependence on others. And perhaps both are needed. In Fisher's (2018) chapter on Google, the co-founders are described as Larry being an introvert and Sergei being an extrovert; together, they made Google a success (to put it mildly). Indeed, entrepreneurship is about valuing and appreciating different skills, it is about knowing what you as a programmer are good at, not being afraid of new things, believing in your ideas, and not being afraid to ask for help.

▼▼▼

As discussed in the previous chapter, tech culture has many influences. In this chapter, I addressed some of its contradictions. Programmers seem to navigate these contradictions by exercising their ability to respect differences and different areas of expertise, coupled with a curiosity to learn and embark in new fields and a pleasure of exploration. Detailing the contradictions in tech culture allows me to begin approaching norms and values in the culture, such as the belief in information and the power of computers to change life for the better, to imagine a new and brighter future and make this a reality through technology. But before I attend to the core of tech culture in Chapter 7, I first attend to the outer layers. Hence, in the following chapters I depart from the analytical framework presented in Chapter 2. Indeed, another way to approach core values of culture is to attend to its symbols, labels, role models, and heroes, which are the topics of the next chapter.

Chapter 5

The geek genius among beanbags & unicorns in Lego land

According to Geert Hofstede (1991), symbols and heroes are outer manifestations of a culture, and thus easier to observe than its core rules and values. Gideon Kunda (2006) further underlines that signs and symbols are expressions of culture. In this chapter, I explore the most apparent signs and symbols that I came across during my journey into tech culture. I start with the emblematic beanbag, a must for any tech conference or tech company with any kind of grounding in the culture. I also discuss a mythical and quite ambiguous creature: the unicorn. Then I turn to Prometheus from Greek mythology as a label. When it comes to heroes, an intriguing result of my interviews was that a majority of the programmers claimed to not have any heroes – they were their own role models. And who are these role models? Geek geniuses. I conclude the chapter with another surprising finding: almost all of my interviewees played with the Danish plastic construction toy Lego as children. Why, you might ask? Because they loved picking things apart and putting them back together. The reasoning about such tinkering and the importance of Lego in tech culture serves as a bridge to the next chapter on rules, rituals, and practices.

Beanbags

Adam, the programmer I met in the old university town of Lund, Sweden, talked about the Öredev conference in Malmö as a venue for inspiration and new knowledge – I decided to go to it. It was a grey and chilly November day in 2018, but inside the conference venue it was nice and warm. The first thing I noticed were numerous beanbags lying around the open areas. I found them to be very comfortable when sitting (semi-lying) and observing the people and activities around me. I noticed an old-fashioned popcorn machine, full of freely available popcorn, and wondered why media and communication research conferences don't have offerings like this. When the first day of the conference came to an end, people started talking about beer: "When is beer o'clock?" Sooner than I thought, apparently. Before the last keynote, rows of beer were

lined up on the tables in front of the main auditorium, and we were allowed to bring one with us into the last presentation of the day. For this occasion, even more beanbags were placed in the large conference hall.

Attending tech conferences was far more comfortable than attending traditional academic conferences, much thanks to the beanbags. Anywhere I went, there were beanbags. I found myself deeply sunk into a beanbag chair not only at Öredev2018, but also at SXSW2019 and the React Day conferences in Berlin. In Bangalore, I even found myself in a store called the Crazy Beanbag. And this wasn't the only beanbag store in this Indian tech metropolis; I also stumbled across the eco-friendly beanbag store Urbanloom, which branded the beanbag as not only comfortable, but also good for people's health. I thought to myself that surely this cannot be the only reason beanbags are so popular in tech culture. Hence, I asked myself, what is this a symbol of, and what does it mean?

I found a clue to the answer in the weird mix of hippie, hacker, startup, and youth cultures. Conceiving of themselves as underdogs – as the young and fun alternative to corporate giants – and rebelling against bureaucracy and authority, programmers are people who mostly don't want to live under an oppressive organisational structure. What could be a better symbol of this than the loose and comfortable beanbag? As with tech culture's rejection of formal office attire, the beanbag represents a general opposition to stiff office landscapes and regular working hours. Anyone who has visited the headquarters of Facebook or Google know that they are anything but stiff.

Beanbags: "Counterculture comfort"

According to *The Silicon Valley Revolution* documentary (Tenhaven, 2017), beanbags can be dated back to tech conferences already in the 1980s. Most known for the beanbag is the Xerox Palo Alto Research Center (PARC), which opened on 1 July 1970 on Stanford University grounds. Fred Turner (2006: 118) describes PARC as elite, technocentric, and self-sufficient, with researchers seeing themselves as exploring the technological frontier as well as "taking on the cool of the Pranksters". This didn't always fare well with the mother company. While corporate Xerox was populated by men in three-piece suits, Xerox PARC was "all about beanbag chairs and hippies riding bikes and wearing sandals" (Alva Ray Smith, in Fisher, 2018: 46).

While in Silicon Valley to conduct interviews, I visited the Computer History Museum, where I read that the beanbag is an "icon of counterculture comfort"; I also got the opportunity to look at a replica of the first beanbag, made out of saffron-yellow corduroy in true 70s-hippie nature (see Figure 5.1). Apparently, the computer science lab at Xerox PARC used them in the room where they held their weekly meetings. In a recorded interview, Adele Goldberg (one

Figure 5.1 *Replica of a Xerox Parc beanbag*

Source: photo by Jakob Svensson

of the collaborators at PARC) shares that Xerox PARC was like a playground (perhaps not so surprising given tech culture's connection to middle-class youth culture). Goldberg also explains that the purpose of the beanbag was to be so comfortable that it would be hard to get up and attack each other. However, according to Goldberg, who was pregnant when she worked there, "what [they] didn't understand was, once you sunk in and you were pregnant, you couldn't jump up. Other people had to bring you up" (CHM, 2011). This might be another example of the "bro-ishness" of tech culture; being centred around young men, the very thought that colleagues could be pregnant – or simply older – doesn't seem to have crossed many minds. Viktoria in Stockholm, for example, told me she avoids beanbags, not because they aren't comfortable, but it's just too hard getting up from them with hip joint issues.

The idea of startup company life is often coupled with irresponsible fun, with companies spending exorbitant amounts on games, food, and brightly coloured beanbags to improve company atmosphere and creativity (Kozulin, 2016). This can be traced back to certain entrepreneurship ideals. Employees' attitudes in the workplace – being almost married to the company – is what makes it innovative, that employees are so involved in the business that motivation isn't a problem (see Girard, 2009). If the basic necessities are taken care of by the

company, then employees can focus on creativity, productivity, and innovation. Google, a pioneer of startup work culture, providing free food, healthcare, and multiple amenities, exemplifies this.

The idea of enabling workers to focus on creative solutions rather than practical arrangements related to everyday life was apparent during my visits to Silicon Valley tech giant headquarters, where I found everything from bike repair shops to well-equipped gyms and spacious ice cream parlours. Google famously allows their employees to devote 20 per cent of their time to side projects of their own choosing. According to the online entrepreneurship magazine and newsletter *Inc.*, this is one reason why Google remains one of the most innovative companies in the world (Robinson, 2018).

The beanbag is also about embracing flexibility, in which not only furniture, but also staff can be easily moved around. This is indeed the case for young Sunil, who now works in his company's branch in Berlin, Germany. Sunil told me he doesn't exactly know when he will be able to return to India. Similarly, Gabriela, the junior programmer I interviewed in Sao Paolo, explained that she is moved around to teams where her coding labour is needed the most. She believed that when she has a position higher in the company hierarchy, she will be placed in a team more permanently. This workforce flexibility, especially for junior programmers, was also confirmed by Sören at Microsoft in Copenhagen: "When you start here, you need to show you can adapt to different teams and work processes". While annoying for Gabriela, many of the freelance programmers I met seemed to embrace the flexibility of being able to choose projects as well as working hours.

Unicorns

One of the most prominent symbols I came across during my journey into tech was the unicorn. Is there anything more suitable for symbolising a better and more enchanted future than the unicorn? I last saw one at a workshop on re-humanising automation, in which the Chilicorn Fund flashed by. Chilicorn works with non-profit organisations "for a slightly better world" (Chilicorn Fund, 2018). Their logo is a white unicorn with a rainbow-coloured tail and mane, and instead of a horn it has a red chili fruit (see Figure 5.2). Why a chili? Because the open source and social impact program it is built around is labelled Spice Program. This is an example of the playful side of tech culture, but also an example of the belief in a better world through digital technology. The program and the fund are sponsored by the digital innovation bureau Futurice, which claims not to predict the future (which is kind of refreshing), but to define it (which is perhaps a bit megalomaniac) (see Futurice, 2020).

Figure 5.2 *The Chilicorn Fund*

Source: Chilicorn Fund, 2018 (reproduced with permission)

During the Middle Ages and the Renaissance, unicorns became allied with reports of exotic creatures and had a reputation of being elusive – things that could be neither seen nor caught. Unicorns were simultaneously described as wild, strong, and powerful creatures of the woods and symbols of purity and grace. This might seem to be an oxymoron, but it resembles how programmers in my interviews described code, or *elegant* code, to be precise. For example, I asked Pelle in Stockholm what he meant by elegant code:

> Elegant code is simple code – simple but beautiful – yet does what you expect of it, if not more. It is strong, as other programmers will understand it and use it in their programs; […] you get it immediately just by looking at the code. It is not banal. There is not a lot of extra lines just to explain the functions. The labels are smart, to the point, and yeah… you just get it.

When trying to create the future with the help of programming languages, the perfect code might be as elusive as a unicorn. Programmers think they might have it, but then find themselves continuously debugging. Scott Rosenberg (2008) illustrates this in an in-depth account of a development software team who never quite made it work. Indeed, programmers spend much of their time debugging, yearning for programs "that will respond less rigidly to the flow of human wishes and actions" (Rosenberg, 2008: 1). As Vikram Chandra (2013) underlines in his highly original book on writing code, beautiful code is lucid, easy to read and understand. As discussed in the previous chapter, code must be recognised and understood by other programmers. Like music, it should be based on harmonious pattern-making and abhor cacophony; excessively frugal engineering with no discernible structure should be avoided. Reading Chandra, it becomes clear that elegant code means *bug-free* code. But as he himself underlines, this remains distant, referring to a study which estimates that most software written today has a bug in every three to five lines.

Unicorns: High status and tamed beasts

Unicorns have been found in seals from the Indus Valley, where they are believed to be a mark of high social rank. Considering the unicorn as a metaphor for code, elegant and bug-free code is thus something that can give programmers high status in the community. "I really like this guy; his code is amazing", as Sunil phrased it. Indeed, it is through their merits – their code – that programmers should allegedly obtain status in this community.

In medieval Europe, unicorns were referred to as ferocious beasts that could only be tamed or captured by a virgin. In a well-cited allegory of the Virgin Mary's relationship with Jesus, a unicorn is trapped by a maiden (supposedly representing the Virgin Mary), and the unicorn immediately lays his head on her lap and falls asleep. The unicorn is a representation of the incarnation of Christ (see History Today, 2019), but, with its attraction to a virgin, it can also represent chaste love and faithful marriage. According to Juliette Wood, author of *Fantastic Creatures in Mythology and Folklore* (2018), a lover is hunted by his love of the Lady and, like the unicorn, he wins her and becomes tamed as a husband. This makes me think about how often speakers at tech conferences mentioned that they had a family or a partner (see Chapter 4). Here, rather than the code, programmers themselves are seen as unicorns, perceiving of themselves as being strong and ferocious, having been tamed into mainstream heterosexual love.

In entrepreneurship culture, unicorns are no myth; they do exist, and increasingly so. A startup company valued at USD 1 billion or more is considered a unicorn. The term used in this way first appeared in 2013 in a TechCrunch blog post (Lee, 2013). Though the post did not explicitly explain why the term unicorn was used, it was probably because this type of company was a rare breed at the time, with only 39 such companies registered in the US. Today, unicorns are everywhere. A CB Insights database revealed a total of 475 unicorn companies as of June 2020; this increased to over 700 by June 2021 (for numbers updated in real time, see CB Insights, n.d.). Narendra, an enthusiastic young programmer I had the opportunity to talk with after a sweaty meetup in a small windowless room in Chennai, India, dreamed of starting a company that one day will become a unicorn. In Bangalore, Vikas brought me to a meetup with the theme, "How to turn a startup into a unicorn". Afterwards, Vikas explained to me that making it to the unicorn level is the ultimate proof of success.

Unicorns are also increasingly common in popular culture. In a piece in *The Guardian*, journalist Alice Fisher (2017) spots unicorns everywhere, from children's television programming to Gay Pride marches. Indeed, unicorns no longer need to be lured from magical forests by pure maidens; it is possible to buy one in the nearest toy store. Fisher notes that it is possible to buy a T-shirt for the Unicorn Fan Club at the low-cost clothing store H&M, get a Unicorn

Frappuccino at Starbucks, and eat Kellogg's Unicorn Fruit Loops (with the tagline, "Magic Has Landed"). In India, I spotted a Honda CB Unicorn motorbike flashing by on the chaotic Bangalore streets. And a colleague of mine even posted on Facebook that it's possible to buy a unicorn taco holder to "make eating pure magic" (see also Pyper, n.d.). Also, in contemporary pop culture, the unicorn has a magical allure to it. It's interesting how this male-coded ferocious beast has become pink and purple and quite "girlie" in children's merchandise. Parents of young children might be familiar with My Little Pony's purple pegasus–unicorn crossover character, Princess Twilight Sparkle, fittingly the main character of the series *My Little Pony: Friendship is Magic* (Thiessen et al., 2010–2019).

The unicorn is also prominent in the gay community (as also evident in Chilicorn's rainbow mane). When I complained about feeling lonely after breaking up with my boyfriend to Kalle, a gay friend of mine from Finland, he responded that I would be fine, because the "unicorns are protecting me". He decorated the message with a unicorn emoji. According to a *Gay Star News* article, "unicorns are cute-magical-awesome-rainbow creatures and this is why the LGBTI community identifies with them" (Wareham, 2018). They are supposed to represent otherness, freedom, and the ability to transform; they are not quite of this world, and their existence seems to blur the lines between what is real and a more enchanted future. Being outsiders (nerds and geeks) as well as underdogs, programmers can easily subscribe to these themes. Indeed, I met quite a few programmers who were gay. When asking them about the link between their sexuality and their choice of profession, to transform into something else online was sometimes mentioned. Kim – a trans programmer in San Francisco who had retrained from the arts and now worked in tech – elaborated on this:

> I just love to create things, think about how I want something to work, and then make it happen. And here I am not being constrained by my biology or something else; it's only my imagination that limits me. And believe me, being trans indeed requires imagination when living in a heterosexual world so utterly defined by only two genders.

According to the article in *The Guardian* (Fisher, 2017), unicorns became prominent during the gay rights protests of the 1970s and 1980s. Today, sparkling rainbow unicorns appear on T-shirts and banners at Gay Pride parades around the world. Dreaming about a better future and making the impossible possible are themes that resonate deeply in tech culture (as I return to in Chapter 7), and the unicorn does connote something rare and magical, a symbol of hope and purity as well as strangeness and a belief that things will get better as long as one believes. For example, in the follow-up to the fantasy tale of *Alice in Wonderland*, the unicorn says to Alice, "well, now that we have seen each other, if you'll believe in me, I'll believe in you" (Carroll, 1871). This is about being open to strange ideas, to the unknown, to the unreal – and that

you have to make a leap of faith to bond with a magical creature. In tech, this magical creature is sometimes code, and sometimes programmers themselves.

Prometheus

According to Greek mythology, the Titan Prometheus stole fire from the gods to give to humans along with the skills to craft with it. This cunning and intelligent crime is considered the origin of human art and science (Wikipedia, 2021d).[1] Tech can be understood in this intersection of art (creativity) and science (logical thinking and mathematics). The somewhat odd combination of art with logic and science was evident in many of my interviews. For example, Jakob, an outspoken programmer at the Daily, told me that programmers needed not only to be good engineers, but also artists, and if they forget this, their "code will be mediocre". Benjamin in Tel Aviv provided another example. I asked him to elaborate on his love for code, and Benjamin answered:

> I think that part of *writing code is very close to working on an art project*, where you, for example, paint; it is like you and the canvas and the subject and everything else fades away. […] You are inside this thought process and you have your full attention on that. Compare that to real life, where you are constantly interrupted, like in academia when you're writing something. It is like those moments when you dive into something, and there is *a little bit of magic* there for me. [emphasis added]

Something that surprised me was how many of my interviewees had a background in the culture section, and mainly in theatre. In addition to Kim, the trans programmer in San Francisco, Benjamin used to be a sculptor. When I asked Benjamin about all the creative people in the business, he stated that when someone is a creative person and interested in making things, the outcome could be a sculpture or a digital platform, and he didn't see programming as any less creative than sculpting. In Stockholm, I interviewed Pelle, who previously worked in various culture projects, particularly in theatre. He highlighted the joy of creativity as a main reason behind his interest in programming. For him, programming required the same creativity as staging a play, only the tools were different. Programming as an artistic endeavour was also underlined by Mark in San Francisco. When I interviewed him over Skype, he mentioned the English musician and producer Brian Eno as one of his role models. This took me by

1. In some versions of the myth, Prometheus and his brother Epimetheus were tasked with providing positive traits to each animal. Epimetheus (meaning "afterthought" or "hindsight") had so liberally bestowed gifts that when it became the turn of humans, there was nothing left to give. Prometheus saw humans struggling, eating raw food and shivering with cold and, despite the gods being determined to keep fire from humans, Prometheus stole the fire and gave it to humans along with the skill of metalwork (World History Encyclopedia, 2013).

surprise, so I asked him why. Mark answered that it had to do with creativity, and that programmers can build beautiful and artistic things out of code:

> I think it is something around creativity and creative flow. And I really see programming as this artistic endeavour and look to artists for inspiration a lot of times. […] When I was young and started programming in C [programming language], I started to look at the code and got a feeling like, wow, it is really possible to build something beautiful, artistic out of code… and I kind of saw the beauty of software and of code for the first time.

This artistic side of programming is also clear in Adam Fisher's (2018) interview book. According to Andy Hertzfeld (presented as one of the software wizards behind the original Macintosh and co-founder of General Magic), besides financial and technical values, there exists a third set of values in Silicon Valley: art values, or the longing to create original things. The ability to create art with a computer is also one of the points in Steven Levy's (1984: 123) hacker ethic: merging art, science, and play "into the magical activity of programming". One of the main reference works in computer science is Donald Knuth's *The Art of Computer Programming* (1968), and, as Rosenberg (2008) correctly notices, Knuth chose to refer to computer programming as an *art* in the title. In the nineteenth century, technology was often referred to as mechanical arts (Peters, 2015), and tech historian Nathan Ensmenger (2015) compares programming to poetry, as does Frederik Brooks (1975) in his legendary book on programming. Brooks rhetorically asks why programming is fun, and one answer he provides is that the programmer, like the poet, is someone who creates art by using their imagination. Programming is therefore considered fun because it gratifies creative longings built deep within.

Chandra (2013), combining his Indian heritage with programming and writing science fiction novels, connects the beauty of code to poetry via Sanskrit rules: language made beautiful through the operations of simile, metaphor, metonymy, double entendre, and so on. The importance of beauty in software design is also emphasised by Paul Graham in his book *Hackers & Painters* (2010). According to him, hackers and painters have much in common, since both are makers, trying to do good things. Chandra (2013) argues that programmers regard themselves as artists, that they care about the beauty of code in addition to its functionality. Hence, code should go beyond the purely practical and aspire to elegance. It thus seems that to become a good programmer, it is important to have artistic qualities in tandem with more science-oriented logical thinking and math skills.

Bart in Copenhagen illustrated nicely how programming is situated in this intersection of science and art. When I asked him if anyone can become a programmer, he explained that not only are programmers problem-solvers, but there is also a desire to do this in in a creative way:

> No, I don't think so… you do need to have a certain mindset… I think you have to enjoy diagnostics a little bit, *you have to have a mindset that is a little bit "sciency"*… you don't have to be a complete math whiz, but *you do need to have some logical thinking*. […] *Weirdly, you also need to be a little bit creative*, which is kind of something that doesn't fit within that profile often… you know you are diagnostic, you are logical, you are "sciency", but when you have a software development problem, you can solve it in a million different ways. You just have to find a way that is *beautiful, elegant, and simple*, and hence coding is a very *creative* process. [emphasis added]

Nadja in Silicon Valley described a good programmer as an artist visualising the bigger picture before starting to "paint the canvas". In other words, programmers first imagine the solution, and *then* begin to code. When discussing with Martin at Facebook what characteristics are needed to become a good programmer, he emphasised that "it is all about thinking outside of the box and being creative". Thinking outside the box was already underlined in Apple's 1997 "Think Different" campaign, believed to be a direct reply to IBM's motto "Think". According to people who were there at the time (see Fisher 2018), the "Think Different" campaign was brilliant because it gave users permission to be different and to feel good about it, and it also changed people's perception of computers as "beige boxes".

In large tech corporations, creative work often has the highest status. According to Kunda (2006: 39) – in one of the few ethnographies of a high-tech corporation – developing new products was seen as glamorous work because this was believed to be "the essence of creative engineering, what engineering is all about". The more creative the work, the higher in the hierarchy programmers were; junior engineers, producing constant streams of code (like Gabriela in Sao Paolo) were the ones being moved around like beanbags.

Much like how the myth of Prometheus illustrates the intersection of art and science, programmers conceive of the creative yet logical practice of coding. But Prometheus also connotes intelligence and cunning. Referring to this understanding of the myth, there's a tale of Steve Jobs's own "Promethean moment". In late 1979, he visited Xerox PARC with a group of Apple engineers and executives. According to some reports, it was on this visit that Jobs discovered the mouse, windows, icons, and perhaps most important, the GUI (Graphical User Interface), which had been locked away at PARC by staff that apparently didn't understand the revolutionary potential of what they had created (Making the Macintosh, n.d.). Hence, Jobs stole the GUI "from Mount Olympus in order to give it to mere mortals" (according to Fisher, 2018: 118).[2]

An interesting aspect about the myth of Prometheus is that he himself didn't need fire; he used his Titan superpowers to help others out of goodness. This

2. This story has been challenged, but it underlines the importance of cunning and intelligence.

resonates with a software tool popular in the newspaper business – mentioned often in my interviews with senior developers at the Daily – the Prometheus software tool for event monitoring and alerting (Prometheus, 2021). The Prometheus software uses code to help programmers diagnose and solve problems. Thus, Prometheus symbolises a helping hand, helping humans with powers out of this world.

The kings (& queen) of Geekistan

Something that stood out in my interviews when asking about role models or heroes was that most of the programmers I talked with claimed to not have any heroes. It can, of course, be argued that the hacker or prankster (see the previous chapter) could be viewed as hero figures; but, as with the lone cowboy coder, these are anti-heroes, lacking classical hero qualities. Instead, programmers referred to something else: their own genius. As Graham (2010: xi) explains in the preface to his book, hackers generally look dull on the outside, "but the insides of their heads are surprisingly interesting places". Or consider how Levy (1984: 66) describes programmer and hacker Bill Gosper as "someone whose brilliance puts things like physical appearance into their properly trivial perspective". In this section, I therefore attend to how programmers approached the figure that was most common in my interviews: themselves.

This self can be described as a geek (or nerd) genius, a brilliant and curious individual with a burning interest and the stamina to pursue his own interests. For example, I asked Andri, an Estonian programmer in Malmö, what he thought the biggest problems are with computers in terms of what he thought they should do that they didn't do. "So far, they have been doing everything I want them to do", he answered with a laugh. When I asked my interviewees to describe their work as if to a child, many of them told me that it was very difficult. "Not even my wife understands", as Sören phrased it, adding "you know, it is a little bit abstract". This resonates in Brooks's (1975) early statement about computers being incredibly complex, the most complex technology ever built. The programmers, however, understand computers. For example, in Lund, I asked freelance programmer Adam how he would describe his work to a child. He was working on educational computer games for kids and he answered that he would tell the children he is "the person who makes the game work". Niklas in Norrköping answered the question similarly: "I am the one that makes it work".

This is also a common theme in the literature. Sara Wachter-Boettcher (2017) writes critically about programmers perceiving of themselves as geeks and boy geniuses. Emily Chang (2018) refers to a geek mythology, a belief that asocial boys obsessed with computers are the best programmers, and few women are

labelled geniuses. According to Chang, this has discouraged women from tech. Kunda (2006: 40) describes how engineers in his high-tech company sorted themselves out, not only by the work they did, but also by their perceived skills: "there are the brilliant and the geniuses". And Graham (2010) argues that there is a strong correlation between being smart and being a nerd. In the title of Fisher's interview book, he refers to Silicon Valley as the valley of *genius*. Following this, when ideas failed, it was often believed nothing was wrong with the idea itself; rather, it was the *users* who just didn't get it, couldn't grasp the brilliant idea right away (see, e.g., Giannandrea, in Fisher, 2018: 176).

I asked Ted in Malmö what advice he would give to a student wanting to enter the world of computer programming: "It is fucking difficult. Either you have this mindset or you don't", he asserted. This hints to a belief that programmers' genius is innate: "You need good intuition", as Sören at Microsoft put it, and continued, "I realised that it was easy for me to pick up computer science". This is echoed in Kunda's (2006: 71) observations that these people are "creative, hard-working, self-governing and can learn". As I discussed in the previous chapter, programmers think of themselves as self-educated or as "just getting it" from the beginning, geniuses that they are. Programmers are supposed to have a natural affinity; hence, in the literature, programming is compared to black arts, and the best programmers are believed to be born, not trained (see, e.g., Ensmenger, 2012; Chang, 2018). At the Daily, Jakob exclaimed that "no one has taught me anything", and Oskar explained that the most talented developers he knew were self-taught. But then, I wondered, what is needed to become a good programmer? Oskar answered that engagement and interest are needed as well as some talent and intuition. Bart underscored this when I interviewed him in Copenhagen:

> I've met so many people the last couple of years when I do my presentations who are *self-taught. They just open a book or a website or whatever*. There is so much information and they just learn it themselves, you know. The thing that you need is the ability to learn. That is the thing that you need. [emphasis added]

The geek genius is thus self-made, a lone inventor "struggling against a dominant corporate dinosaur", or a lonely nerd "turned accidental billionaire", as Ensmenger (2015) eloquently puts it: an underdog who against all odds produces tech that is immediately recognised as revolutionary.

When the programmers I interviewed did talk about education, the story was most often a tale of them acing everything and (or) getting bored by the slow pace and thus learning more things in their free time. Andri, for example, told me how, as a bachelor's student, he just bought books, read them, did some exams, and voilà, he got his Microsoft certifications. Therefore, at the end of his university years, he had, on top of his bachelor's degree, "a big list of certifica-

tions". The geek genius seems to pick up new things very easily, which traces back to the importance of having an interest in something. A geek, or a nerd, is "boringly studious" and "a single-minded expert in a particular technical field" (Lexico, 2021b); therefore, programmers need an interest and passion in what they do. This was highlighted in the majority of my interviews, when I asked what was needed to become a good programmer.

The geek genius is also a doer, someone who prefers to act rather than talk or listen. Daniel, programming the front page at the Daily, complained about too many meetings and micro-management from the top. He claimed that programmers "are doers, even though we don't always know what to do". He was frustrated that sometimes he and his programmer colleagues were just sitting and waiting for instructions. Apparently, too many stakeholders (such as editors, journalists, the subscription department, etc.) had opinions about the front page and its functions. "Let's talk less and just let us program", he stated. Python in Berlin complained about his university studies in a similar way: "There's a hundred and twenty people sitting and listening to someone talking, when you just want to have your fingers on that keyboard". Nadja in Silicon Valley also explained to me that she looked for doers, when answering my question about what characteristics she looked for when employing new programmers:

> I look for an ability to learn, to make decisions independently […]. I think if people should try rather than sit and wait for instructions, they will succeed more, […] *to be doers and not talkers*. [emphasis added]

In the literature, Graham (2010) differentiates mathematicians and hackers in a similar way: mathematicians *study* algorithms and hackers *use* computers as a means of expression. In this way, Graham (2010) underlines hackers as *makers*, wanting to figure out programming by hands-on experiences and not by theoretical studies. In Sweden, we have an illustrative proverb: "a lot of talking and no hockey". Following this, programmers prefer to "play hockey" than sit in endless meetings. One hacker in Levy's (1984: 107) book, for example, refers to others as just "theorizing", while he is the one rolling up his sleeves and "doing it". Also, in his chapter on the Homebrew Computer Club, people are described as not just sitting around waiting for things to happen, but rather doing stuff. From a less celebratory perspective, Evgeny Morozov (2013: 133) describes a general impatience in tech culture, especially with politics, because programmers "think it involves nothing but talk". Levy (1984), on the other hand, labels this as a hands-on imperative, and John Markoff (2015) refers to a maker, or do-it-yourself, mentality. While Nadja complained there were too few doers in Silicon Valley, according to computer engineer Orkut Büyükkökten, it is still easier to turn ideas into reality in Silicon Valley than anywhere else (in Fisher 2018). And for this, programmers must be doers *and* thinkers at the same time (according to Steve Jobs, in Fisher 2018).

In my interviews, when pushing about the question of their role models, that there might be a person at work they looked up to, Sunil answered: "I look up to people in the office who are working with me, how they solve problems". Or, in the words of Benjamin:

> I don't have a hero, I don't admire public figures, but I do look up to many of my colleagues who are brilliant and have tons of experience and knowledge. And this is inspiring, as you are constantly learning.

The admiration of problem-solvers and knowledgeable colleagues can be connected to the norm of not striving too much for attention (as attended to in the previous chapter). For example, at a meetup in Cupertino, some attendants weren't happy with the organisers' wish that we introduce ourselves with our titles: "I'm just a guy", as one attendee rebelled when it was his turn to present himself. A programmer is a geek, a curious tinkerer who prefers to work alone and not seldom starts exploring technology out of boredom combined with some kind of innate talent for tech. Jakob, for example, talked about how his dad brought home a computer and how all his friends were playing games on it. But, according to Jakob, these early games were "worthless". "So then I started to explore the game instead. I looked at the code to see how it functioned and then improved the game", Jakob told me.

While still male-coded, the geek is different from the dude.[3] Pelle pointed to this when explaining why he took a break from programming to work in theatre (before returning to programming); he wanted to expand his social circle (I also got the feeling he wanted to meet girls). I asked him whether his group of friends who used to code together when he was young were all boys. Indeed, this was the case. But, when I asked if programming was kind of a "dude-ish" thing, Pelle stressed that a geek is a different kind of masculinity:

> So… it wasn't "dude-ish" if you say so, but more interest-driven. […] We were a bunch of guys who had the same interest… because that's interesting… because I think no one thought we were very masculine, you know. *It wasn't macho but more geek*. And if there would have been a girl there, it probably wouldn't have had anything strange about it… and then I don't know… it's a safe little world, to be lost in programming, you know… it was our interest, so it was a bit geeky, or a reality escape, perhaps. [emphasis added]

It's a wonderful quote, as it is possible to sense a kind of ambiguity or contradiction of the geek genius not belonging to the cool guys (being an underdog) and preferring to stay out of the limelight and also seeking to escape reality.

3. Chang (2018) also elaborates a development from what she labels the nerd to the bro.

In Lego land

An interesting aspect of my interviews was that a large majority of the programmers I talked to played with Lego as children. I asked Anders, a middle-aged programmer at the Daily, how he became a programmer, and he explained it was just an extension of his playing with Lego as a child: "It's always nice to pick things apart, so to speak, and then put it back together". This curiosity was often coupled with a quest to try to understand how things worked. Python remembered how, after some time playing games on his first computer (a Commodore 64), he got curious about how the computer worked, and that was how he was eventually drawn to programming. Roger at Google talked about his childhood with friends coming over, and they would pick things apart and put them back together. I asked him whether this involved Lego, and his answer was telling: "Oh yeah, *oh yeah*… I had everything; my Lego was my most prized toys" [emphasis original]. If programmers didn't have Lego, they picked apart something else.

In my interview with 50-something Nadja, she told me that, as a kid, she took her transistor radio apart and then put it back together. Bart in Copenhagen remembered his childhood and recalled how every time they had something electronic at home, he liked "to rip it apart and put it back together", sometimes to his parents' horror. Similarly, Andri in Malmö talked about being ready to "tear everything apart and understand what is in there". Levy (1984) similarly describes early hackers as tearing apart old television sets, or a radar disk, and from its pieces rigging a parabolic reflector. And Fisher (2018) writes in the preface of his book about his own fascination with taking apart his computer.

The love of taking things apart and putting them back together hints at an important skill in programming, namely that of logical thinking, breaking a problem down in smaller units, and then putting these units (or lines of code) back together. Some programmers also referred to this as modular thinking, or modubility, a concept Andri tried to launch when describing the ability to work and think in terms of modules. To think in terms of steps, or decision trees, is basically what algorithmic computation is about; if this step, then this (as highlighted in the title of Bucher's 2018 book on algorithmic power, *If … then*). Pelle in Stockholm talked about an ability for abstraction – "to see the big in the small and the small in the big" – or to be able to think about problems in terms of layers or modules, and how these could then be fitted back together. Talking to programmers at the Daily about when they automated the news ranking on the website, they remembered that discussing a whole web page was too big, so they broke it down in smaller modules to make the problem more manageable. Algorithm programming is defined in a similar manner in *Algorithms for Dummies* (Mueller & Massaron, 2017: 30): "You must break down the problem

completely, define precisely what you need, and then, after you understand the problem thoroughly, use the correct steps to find what you need".

This was sometimes discussed as engineering. Anna in Oakland talked about her dad as an engineer at heart. She remembered when he got his first computer, how it was like Christmas for him every day: "I mean, he could spend an entire day just building a spreadsheet… and he was just happier than anyone". Engineers want to build things, and this is why, according to Roger at Google, it is difficult to get engineers to stay on a so-called legacy project for long. Why? Because they want to build something new and interesting, to create something rather than just maintaining or managing something already built. Kunda (2006: 40) refers to this as an engineering mindset, arguing that this is why those who worked in management and were not directly connected to the problem-solving were derogatorily labelled as "overheads", "do-nothings", or "product preventers" by the programmers in the company he studied. This is about "playing hockey" rather than just talking. And if it is an engineering problem, it can be solved. I asked Nadja if she thought everything could be solved with code, and she answered that it is possible to solve pretty much every technological problem with engineering. What programmers cannot solve, she argued, were the human problems. Benjamin in Tel Aviv highlighted the importance of being analytical in a similar manner:

> You need to be able *to break down a problem* and to think about the same problem from different angles, to be able to hold very complex ideas and systems in your head. You need to have a tendency for mathematics, space and geometry, and things that are analytical. *Like playing with Lego*, picking things apart and putting them back together. [emphasis added]

This theme is present in the literature as well. Chandra (2013: 123), in his outline of beautiful code, talks about small coherent sections fitting together seamlessly, "like pieces of complex mosaic". He explains how programmers must "chunk the system" to allow for variation, a kind of "assemblage of functionality", logical and orderly (Chandra, 2013: 125–126). Physicist Max Tegmark (2017: 77) explains how "computers gain efficiency by splitting their work into multiple steps and reusing computational modules many times".[4]

Rosenberg (2008) has a whole chapter in his book on what he titles "Lego Land", referring to a dream – the Lego Hypothesis – where programmers

4. I have thought about this as a Lego fallacy, that all programming (and hence also artificial intelligence) would be a rearrangement of smaller units. Tegmark (2017: 284), for example, believes that humans are "simply food rearranged". If picking humans apart like Lego bricks, the mystery of human consciousness could be solved, and perhaps humans could be put back together in a better way. This is also how Tegmark approaches qualia, the term for individual instances of subjectivity and conscious experience in philosophy and psychology (such as the perceived sensation of happiness when the sun shines, or the spicy taste of Indian food) as basic building blocks of consciousness. It is in the arrangement of the parts that magic is supposed to happen, how the pieces are put back together.

would have all the code they needed, and programming would just be about assembling lines of code and software parts in a new order to achieve whatever a programmer wanted to achieve. Software engineers would then be "free from the mundane necessity of programming" (Rosenberg, 2008: 94). If only software was like Lego: small, indivisible, and substitutable. However, this Lego dream has not been fulfilled. One reason, Rosenberg (2008) argues, is because programmers are motivated by reputation, and thus feel a certain pride of authorship (as also discussed in the previous chapter); to just pull parts off a shelf and reuse them would not satisfy programmers in this regard. The urge to pick things apart is not the same as an urge to put another programmer's code back together.

Taking things apart to see how everything fits together and rearranging modules can be summarised with the practice of tinkering. Ensmenger (2015) describes tinkering as a highly specialised puzzle-solving that not only requires skills, but also experience and genius, and Levy (1984) connects tinkering to curiosity. Turner (2006) concludes that major technological innovation has arisen from individual tinkering. The programmers I interviewed talked about tinkering as trying out new solutions and seeing the consequences, or in the words of Andri in Malmö, "if I tinker something here, what are the consequences there?" This can also be related to bricolage, defined as tinkering and innovation, according to Thomas (2002). Indeed, bricolage can be understood as a style of rearranging elements of existing cultures to create something new. In art and literature, the term refers to construction or creation from a diverse range of available things.[5] Sherry Turkle, in her pioneering book *Life on the Screen* (1995), uses the concept of bricolage to describe a particular form of problem-solving in programming, and she even titles one chapter "The triumph of tinkering". She contrasts modernist top-down planning with bottom-up bricolage: "problem-solvers do not proceed from top down […] but by arranging and rearranging" (Turkle, 1995: 51). A bricoleur arranges and rearranges well-known material, as with Lego. Turkle connects this to anti-authoritarianism, rearranging someone else's construction, as well as a desire to play.

▼▼▼

Symbols, labels, and hero figures are outer manifestations and expressions of culture that point to core rituals, practices, values, and imaginations. In this chapter, I presented how the beanbag symbolises flexibility and underlines a preference for the playful, as well as an underdog position. I then turned to the unicorn, which is used in a plethora of different ways, but all coming back to

5. In a sense, what I do now, writing this book, is a form of bricolage, taking parts of existing literature and combining it with parts of my interview transcripts and observation notes, and putting it together in this book.

how the mythical, magical, and impossible can be realised through a programmer's creative imagination, whether it is perfect bug-free code, a better future, a loving relationship, or the ability to transform in a normative society. The Prometheus myth ties science and art together and provides an apt description of programming that values both creativity and logical thinking at the same time. This further connects to the hero figure of the geek as a doer, someone who makes things happen, not seldom by picking things apart, figuring out how they work, and putting them back together, sometimes creatively rearranging the parts in a different way just like they did with their Lego as kids. My interviewees referred to an urge to understand how things worked, or didn't work, and if their code wasn't working as anticipated, they picked it apart to study why. Lego thus becomes a symbol for tinkering, which leads me to the practices and rules of the game, such as iterative data-driven development, which is a topic for the next chapter.

Chapter 6

Mind-blowing flow, grit, & data-driven development

In this chapter, I attend to the rules and practices of programming in tech culture. This can be connected to the concept of logics, understood as sense-making and meaning-generating frameworks for practices and experiences (Asp, 2014; Dahlgren, 1996; Lowrey, 2017). Logics thus organise work and provide stability to social behaviour.

Understanding logics as rules of the game, or sense-making frameworks, is evident in more anthropological approaches to culture in general, and in so-called rituals in particular. Gideon Kunda (2006: 92) writes about rituals in terms of "a rule-governed activity of symbolic character [… which members of the culture] hold to be of special significance". Rituals are thus collectively produced and create a shared definition (or frame) of situations and may therefore determine how situations are perceived, interpreted, and understood. The rituals Kunda observed in his ethnography had two distinct features: 1) a decentralisation of power, and 2) an embrace of informality, openness, and individual initiative. These rituals directly relate to a general anti-authoritarianism in the culture.[1]

Continuing the discussion from the previous chapter about tinkering with Lego, I first attend to data-driven development. Then, I discuss the state of flow, followed by a section on the importance of having so-called grit, which is connected to a belief in oneself as a programmer and never giving up on one's ideas and solutions. I conclude by discussing the iconic demonstrations of new technology and new tools that are supposed to "blow our minds away". Rules and rituals that I don't directly attend to here are the importance of informality and sharing, underlying, for example, the Homebrew Computer Club in Menlo Park and the open source movement. The importance of having fun, doing things "for the lulz", and the generally prankish and playful side of tech culture have also been attended to in previous chapters.

1. It's interesting that such rituals took place in a high-tech corporation, which is more hierarchically stratified than smaller tech companies and freelance programming.

Data-driven development

At the Computer History Museum in Mountain View, I read that programming is about making error and recovering from error. Indeed, debugging is a major part of the programming practice. Steven Levy (1984) describes how hackers spent full days debugging and, in the book *Dreaming in Code* (2008), Scott Rosenberg followed a team of software developers during three years in which bugs take centre stage. This is revealed already in the subtitle: *Two Dozen Programmers, Three Years, 4,732 Bugs, and One Quest for Transcendent Software*. It was Linus Torvalds and the open source movement who introduced what Rosenberg labels a bazaar style of programming, which I connect to a rule of data-driven development in tech culture. The tedious work of debugging can be outsourced as software programs are rolled out little by little. Data-driven development is about releasing software piece by piece, getting feedback on how users react and engage with the software, and improving and adapting the software when things go wrong. I think of my computer, or phone, which continuously wants to update itself to improve some features (often happening at the worst moment, when I need my device the most).

Data-driven development can also be connected to speed – to do things *now* – and is about prioritising immediacy over quality. "Just shove it out there and see what happens", as Katie Geminder (Facebook's first project manager) phrases it in Adam Fisher's (2018: 361) interview book. Entrepreneurship and startup culture also value the fast and the disruptive, which is apparent in my empirical material. "Be a speed understander, become good at seeing what other people miss out on", a presenter at SXSW2019 asserted. Sam in Silicon Valley put it like this: "Maybe something that happened last month is outdated this month, so you need to move fast". Python in Berlin explained that "you can't be good and fast at the same time", and programmers must decide whether to be "finished on time, or deliver quality". In other words, programming is *not* about doing it right from the beginning. Of course, programmers want their programs to be perfect, but, according to Levy (1984), no program ever is – but this has never stopped a hacker. Paul Graham (2010) similarly argues that just like painters, writers, and architects, hackers figure out programs as they write them. Producing a top-notch program is about gradual refinement, and, just like an oil paint "a good programming language should […] make it easy to change your mind" (Graham, 2010: 27). It's rare to get things right the first time; there will always be bugs. Indeed, finished software seems to be as elusive as unicorns. A common joke, displayed at the Computer History Museum, is that software isn't finished until its last user is dead.

Since speedy delivery and piecemeal roll-out of a program imply that code probably won't function perfectly immediately, bugs are expected. Rolling

software out little by little, getting feedback in terms of data and improving the code accordingly, and not being afraid of things going wrong is epitomised in Facebook's now famous motto: "move fast and break things". And there is indeed a tendency to equate quality with speed, "the computing machine is good because it is fast", as Norbert Wiener (1948: 125) argued. Almost 70 years later, Frank Pasquale (2015) laments what he labels an addiction to speed, because of the collateral damage that follows when everything must be fast, highlighted by Amazon CEO Jeff Bezos's stance: "We invent, and then deal with the problems later" (Jacoby et al., 2020). Indeed, the bias of automation is towards speed and fast-changing environments, as media scholar Mark Andrejevic (2020) puts it in a more recent account.

Moving fast and not being afraid of breaking things requires a certain kind of boldness. According to computer programmer pioneer and US Navy Admiral Grace Hopper, speaking from a virtual display at the Computer History Museum, "a ship in port is never safe", and it is "easier to ask for forgiveness than to ask for permission". The urge to work quickly and hands-on is connected to a general dislike of slow bureaucracies and governments. There is also a connection with how many of my interviewees look back at their time in school; lectures, they said, were boring, and the questions they raised were seldom, if ever, answered quickly enough. But it is also connected to a general belief in themselves and their ideas – to show that they are able to solve any kind of problems – as well as to the hacker mentality of doing things because it is *possible* and *fun*. Therefore, programmers don't always know from the beginning whether or how their code will be useful. Apple's future analyst Marc Porat, for example, states they will continue to "push technology until someone figures out what to do with it" (in Fisher, 2018: 422). The story of Google is another example. Larry Page and Sergei Brin didn't really know they were creating a search engine. They did things because they could, not knowing exactly what it could be used for, and then little by little creating one of the biggest business successes of our time (see Fisher, 2018).

Speedy delivery and rolling out little by little imply an ambivalent relationship to time in computer programming. In Frederick Brooks's now famous book on software engineering and project management, *The Mythical Man-Month* (1975), he argued that coding, like cooking, takes time; the better the food, the longer it takes to cook. A man-month is a hypothetical unit of work representing the work done by one person in one month. However, Brooks concludes that adding human resources to a late software project in order to get it done faster, in fact results in even more delays; new team members must get up to speed before they can even start working, which often means that existing members are drawn away from active work into training roles. This sentiment – "adding manpower to a late software project makes it later" – came to be known as Brooks's Law.

Rather than the number of people in a project team, the important thing in data-driven development is the feedback loop. More than 70 years ago, Wiener (1948: 110) underlined the importance of the feedback loop in his theory of Cybernetics, the transmission and return of information, something he labelled "a chain of feedback". Decades later, data-driven development was still a central theme when I interviewed programmers. Daniel, in the programming team at the Daily in Stockholm, for example, explained that programmers always had to work with what he called "code maintenance". Sören, the confident programmer at Microsoft Copenhagen, claimed, "in program development, you are stuck with what you do. If we release something, we need to support it for five years". Code maintenance and support isn't an entirely joyful activity, apparently, as illustrated by, for example, Rosenberg (2008).

The feedback loop can, however, also bring about joy. Python in Berlin claimed that the instant feedback he could get explained much of his love for programming, that if he started the program, he could immediately see if it worked or not: "There is this immediate feedback loop and that's what I really like". Pelle in Stockholm also related the feedback loop to the breaking of problems into smaller units (think Lego) and then trying to solve them little by little: "If you take a problem and chop it into pieces, then the problem becomes smaller and not that overwhelming". According to Kristina, the Polish programming student in Berlin, the "*right way* of solving a problem [is to] break it into small pieces and then try to solve it step by step, and then test if it works [emphasis added]". Similarly, Per, a confident older programmer heading a development team at the Daily, talked about his job as a continuous process, where his team built the platform and then changed it depending on the feedback they got from their users (in their case, readers). He referred to the process as "learning by rolling out": his team made improvements and then went live, this would generate data they could use to measure how the site was performing, and they would make improvements, and so on. The key, Per argued, was to decide in which direction to go and then proceed "iteratively, little by little". In this way, programmers are both rationalists and empiricists, emphasising logical thinking while also eager to test and see what happens (see also Domingos, 2017).

The feedback that programmers act upon in data-driven development comes in various forms of *data*. Data receives much hype nowadays, and being data-driven and practicing data-supported decision-making undoubtedly has a positive tone to it, even though in academia there is a whole area called Critical Data Studies, as asstended to in Chapter 1. Programmers at the Daily talked with pride about working with a "data-driven product" (here, a newspaper), and using data to support decisions of how to ameliorate a news-ranking algorithm by finding the right type of data to support decisions about how to tweak and maintain it. The Daily had a whole data analytics team exactly for that purpose: to support the development and end product with feedback data.

Data-driven development is also apparent in so-called Agile programming. Agile is a term used to describe approaches to software development that emphasise incremental delivery, team collaboration, continual planning and learning, and gathering and implementing feedback continually, instead of trying to deliver all at once at the end of a project (Visual Paradigm, 2020). According to its manifesto, Agile advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change (Beck et al., 2001). When visiting Bangalore, I attended a meetup for programmers interested in Agile. The enthusiastic facilitator embraced the metaphors of being agile, quick, and flexible. After the meetup, I met Nilesh, who was inspired by the talk. Being young, willing to learn and adapt, and with a fresh master's degree in his pocket, Nilesh clearly saw his future in programming:

> This is how India will become leading in IT, because our programmers are younger, quicker, and able to work longer hours… and we don't just talk about how programs *should* work, we test and then we improve. [emphasis original]

Here, the principles of Agile resonate in the Swedish proverb I introduced in the previous chapter, in which hockey (i.e., to do things) is more valued than spending a lot of time just talking. Or, as Mark Zuckerberg (2012) argues in The Hacker Way:[2]

> Hacking is also an inherently hands-on and active discipline. Instead of debating for days […] hackers would rather just prototype something and see what works. […] Hackers try to build the best services over the long term by quickly releasing and learning from smaller iterations rather than trying to get everything right all at once. […] The Hacker Way is an approach to building that involves continuous improvement and iteration. Hackers believe that something can always be better, and that nothing is ever complete.

Apparently, Zuckerberg had "done is better than perfect" painted on Facebook's office walls, illustrating that code and software programs are always in progress and never quite finished. My interviewees also underlined the importance of characteristics such as rational thinking, logical thinking, and picking things apart in order to understand how they function. They sought mastery and control of technology at the same time as they cherished the trial-and-error kind of approach. Per at the Daily, for example, described their news-ranking algorithm as a living thing: "It should not be static". He continued to define himself as someone who tweaks, who makes sure to improve the algorithm continuously.

2. I have rearranged Zuckerberg's statements here to make my point; in the original they are in the reverse order.

When journalism clashes with data-driven development

Journalism researchers Seth Lewis and Nikki Usher (2013) refer to iteration (continuously releasing unfinished code for beta testing) and tinkering (privileging play and participation) as a contrast to journalists' way of working. At the Daily, journalists follow strict rules of deadlines, when everything must be done and ready to be published, in contrast to programmers' preferences: "A news article you can write in less than an hour, a new web function can take up to three weeks to develop", as Jonas in the web development team phrased it. Viktor, a programmer on the floor, complained about journalists coming to him at 3 o'clock in the afternoon asking for a function to be finished by 5 o'clock. According to him, this is among the largest conflicts between programmers and journalists. To release a whole new web page at once – and with a strict deadline and a lot of time pressure – "is terrifying" and doesn't suit the programming way of doing things. Jonas told me about how the top management were keen to decide on a date in order to book advertising space in the subway to announce the release of the new front page. "As a developer, I like to have goals to work with, but I don't like that on this specific day, we're going to wallpaper the whole subway announcing the release of the new web page", Jonas said.

Interestingly, one of the editors, Stina, who was also involved with the development of the news-ranking algorithm, refered directly to the proverb about a lot of talk and no hockey, though she said it can be "too much hockey sometimes". But she acknowledged that maybe it's better to be at "the forefront" instead of "engaging in infinitive investigations". The importance of being at the forefront does indeed resonate in the entrepreneurial and startup side of tech culture. And this might be one reason behind programmers' eagerness to roll out immediately to test if the software works. However, the Daily is an old and proud newspaper and not at all like a tech startup, and Stina explained to me that they (referring to journalists in the organisation) must learn that it is good to fail – not an easy lesson for everyone, according to her.

Flow

During my journey into tech culture, I often found evidence of a certain state of mind for programmers: flow. Flow is about forgetting time and bodily sensations and being immersed in the problem at hand: "You totally forget about time, you look at the watch and it is already 4 pm… shit", as Python in Berlin put it. According to Benjamin in Tel Aviv, it's also important "to have a big ass", referring to a Hebrew proverb about the ability to sit down and maintain concentration for long hours and being able to enjoy it.

Levy (1984) provides descriptions of a state of pure concentration, hackers with their minds almost merged with the computer, a kind of super-human

concentration. At a book release at SXSW2019, this was described as losing track of all time and having an intense feeling of joyous concentration that leads to magical creation. Indeed, it is when "the minutes and hours just swish by, then you have flow", as Daniel at the Daily explained. Programmers thus talked about both *being in* flow, but also *having* flow. When in flow, programmers were creative, they made things, code streamed out of them as they approached some kind of solution to the problem they had set themselves to solve. In this sense, flow has a kind of mindfulness to it. At the SXSW2019 launch, I was told that flow is about "giving birth to creative ideas" by letting them flow "through us, through our pipelines". In a similar way, Fred Turner (2009), when comparing tech culture to the Burning Man festival, describes flow as an extended, ecstatic feeling of interpersonal unity and timelessness.

At the SXSW2019 launch, it was emphasised that one should be aware of and try to avoid and eliminate roadblocks to flow, such as fear and anxiety. We in the audience were advised to quiet our inner critics, rid ourselves of traces of unworthiness and scarcity, and not give into resistance, self-censorship or repression, chaos, grudges, or grievances. In other words, we should believe in ourselves. Flow, we were told, is about "bringing order into chaos, about radical acceptance, releasing resistance, total self-expression, total trust, wholehearted engagement and love". Some of tech culture's roots in hippie culture are visible here, with flow having striking similarities to meditation, that is, full concentration, in which everything else fades away.

I first heard about flow when interviewing Estonian programmer Andri in Malmö. I asked him about his perfect workday and what needed to happen for him to leave work happy:

> Basically, I do know that *being in flow* is definitely something that makes me happy, I know the feeling […] in the Csikszentmihalyi sense of the word… you cannot sense what is around, and when things happen and you solve some kind of problem and you feel ok, I made some kind of difference, something like that. [emphasis added]

I asked Andri to describe "being in flow" a little bit more, whether it is about forgetting time completely or being consumed by the task ahead, or something else:

> That and… what else, it is hard to describe… *you stop noticing what is around*, you stop noticing that as well. [emphasis added]

As Andri hinted, flow is also a concept in psychology. Psychologist Mihaly Csikszentmihalyi (1990) describes flow as an optimal experience, energised focus and complete absorption with the activity at hand to the extent that nothing else seems to matter, resulting in a transformation in one's sense of time. The ego falls away as skills are utilised to the utmost. There are eight main characteristics of Csikszentmihalyi's concept of flow:

1. Complete concentration on the task;
2. Clarity of goals and reward in mind and immediate feedback;
3. Transformation of time (speeding up/slowing down);
4. The experience is intrinsically rewarding;
5. Effortlessness and ease;
6. There is a balance between challenge and skills;
7. Actions and awareness are merged, losing self-conscious rumination;
8. There is a feeling of control over the task.

(Oppland, 2021: "The 8 Characteristics of Flow")

In my other interviews, this state of flow came up many times when I asked programmers to describe their perfect workday. Anders in the tech team at the Daily, for example, described such a day:

> A perfect day is when you are in flow, you see your target, are able to break the problem into smaller pieces, and it all makes sense as you deal with the pieces one at a time and then you approach your target.

Anders thus made a connection between flow and data-driven development, tinkering, and breaking problems into smaller units. If everything comes together, and if programmers approach their target accordingly, flow seems like a very fulfilling experience (compared with debugging). "Flow is extremely lovely", Anders reflected. In Berlin, Python told me that flow is what makes programming fun, describing it as being "a little high, like when you run for ten miles and the body starts producing endorphins". He remembered his extra work at a metal press as a kid, and how time passed very slowly: "Now, it is the other way around; my job is not a burden". Similarly, Benjamin, now in the midst of starting his own company, missed the calming influence of coding:

> There is something about it that is very quiet and calming, and being by yourself in front of the screen, I love the pace of it.

Flow is thus linked to the obsessed nerd, the tinkering genius with an immense interest, and hence willingness, to let oneself be consumed by one's interests and the problem at hand.[3]

3. In early programming, computers weren't personal, but rather confined to university computer labs. Nathan Ensmenger (2015), in his historical account, writes about a quest to get time with the machine, and that it was this quest that would cause these boys (mostly) to neglect their bodies (the so-called computer bum phenomenon discussed previously). Flow describes this computer–programmer love affair neatly; the only thing that matters is time spent with the computer. This is where a programmer becomes one with the machine, with the network, and hence, the universe. Like Henry Dorsett Case, the protagonist of Gibson's (1984) science fiction novel *Neuromancer*, it was when he connected himself up to the Matrix and became part of cyberspace that he felt at ease, at home, and fulfilled.

Ted, the app programmer in Malmö, described his perfect workday as a day when he achieved concentration to the degree that the outside disappeared, and he was "completely locked in to a little bubble". He also told me a story of a colleague who he described as extremely skilled at flow, because he could get into this state very quickly. Ted recalled asking him a question, and how his collegue – half an hour later – answered it; he had registered the question in the back of his head, but wasn't willing to allow the question to disturb his flow until he had finished his task. "That is flow and he is bloody good at it", Ted stated.

It seems as if flow is easier to reach on one's own, without a lot of people or distractions around. According to my interviews, it is possible to reach flow in a group setting, but Andri in Malmö told me he had not had any good experiences of it. "Team flow", as he labelled it, was most likely to occur if programmers were in a team that had a really tight deadline and were working extensive hours together to meet it. Python, working freelance, talked about the problem of working in an open office setting in a similar manner, how he needed to find peace and quiet in order to program:

> It's a challenge being in a big noisy room with eleven other people, you don't get into this flow, on the contrary, you are constantly ripped out of it.

Having a family can be somewhat problematic in this regard as well (in a sense, underlining programming as a young and male profession). To program is so immersive that it is difficult to sustain any other relationship than that with the computer and the code. Andri has a big family and this sometimes makes it hard for him to reach the state of flow; therefore, he preferred to work at night. Night working also has less distractions in terms of chat messages from colleagues. Similarly, Ted described the distraction of having a family to attend to when in flow:

> You don't want go home, but you have to. So, a day like that I go home, cook dinner, fix things with the kids, and then I'm back in front of the computer another five to six hours.

Thinking about Ted's wife and kids, I asked if people in his surroundings could get annoyed when he's in flow:

> Yes, yes, yes. Everybody gets annoyed. Anyone who tries to seek contact with me will get irritated.

Flow is a dimension in which programmers achieve what they set out to do, and "not just responding to things happening outside your bubble", as Benjamin in Tel Aviv phrased it. Indeed, the underlying factor for flow was often to solve a problem. I asked Ted in Malmö if what he did was so interesting that it was worth spending all these hours in front of his computer. He answered that my

question missed the point, that it was rather about focus than interest. According to Ted, "after the kids are asleep and I finally have a revelation and solve the problem", flow has worked its magic. I discuss the connection between flow and problem-solving more in the next chapter.

Grit

In order to find flow – and also to survive in the culture – programmers need *grit*. Grit is understood as drive, strength of character, and not letting oneself be deterred by setbacks. When experiencing obstacles, programmers get back up and stick to their guns. Commonly associated concepts include perseverance, hardiness, resilience, ambition, the need for achievement, and conscientiousness, here understood as being careful or diligent, which requires a great deal of self-discipline (Wikipedia, 2021a). As Brooks (1975: 8–9) underlines, "with any creative activity come dreary hours of tedious, painstaking labor", because "each forward step is matched by a backward one" (Brooks, 1975: 123). American inventor Thomas Edison put it another way: "I have not failed, I have just found 100,000 ways that won't work" (see, e.g., Ruth, 2016).

Grit was a recurring theme in my interviews, especially when I asked what it took to become a good programmer. Nilesh in Bangalore answered that it took "time, commitment, effort", and if it something wasn't working out the way he wanted, "you don't give up… you persevere". Ted in Malmö talked about grit as a characteristic he looked for in new co-workers: "I want people who are 'gritty'". I asked him what he meant by that, and he answered that he was looking for people who were persistent, "and if you fail, you try again, and you don't give up". This is somewhat in contrast to the kind of innate genius discussed in the previous chapter. I asked Ted about people who claim they don't have the talent for maths, and he explained there was no such thing as being bad at programming or being bad at mathematics; programmers just need to practice:

> I don't believe in [not having enough talent]. *You just haven't counted enough, practiced enough. Math is a discipline where grit is needed*. It's just a matter of solving a sick amount of math problems. And if you don't understand, then you have to count more, calculate more… and I think this is a bit related to whether you are good at programming, because then you are good at maths, and then you have grit. [emphasis added]

When I asked him to expand on what he meant by grit, he further distinguished it from talent:

> I don't really think grit has anything to do with talent for programming or maths. But I do like people who have grit, I think that is a good characteristic.

Like flow, grit seems to be connected to the nerd; in order to persevere, programmers must have a deep interest – a passion – for something and set their minds to a particular goal they want to achieve. Ted made this connection as well. Grit must be built upon interest, because "if you are going to have this perseverance and not give up, then you have to be insanely interested in solving the problem at hand".

In the literature, I find many examples of grit. Levy (1984: 354) talks about computers fighting back and the need for perseverance, persistence, and patience, approaching programming with "meticulous compulsiveness". Graham (2010: 28) similarly argues that all art, including programming, demands total dedication; like a Renaissance painting, "great software requires fanatical devotion to beauty". Kunda (2006) describes "the golden bull award" in the tech company he studied, representing the spirit of the bull, putting its head down and plowing through problems. Indeed, grit is connected to commitment, strong identification with company goals, and intrinsic satisfaction from work. As one of Kunda's (2006: 10) interviewees phrased it, "I'm not a workaholic – it's just the place, I love the place". In Fisher's (2018: 212) interview book, there are stories of sleeping under the desk and working non-stop, "to work hard and live on coke". The presence of Coca-Cola is also a theme in Levy's (1984) book on hackers. Grit might explain the preference for sugared carbonated drinks in an otherwise quite conscious culture (remember from Chapter 4 the conference-going programmers' preference for Coke instead of filling up their eco-conscious water bottles with tap water); programmers need sugar to be able to code all night.[4]

The focus on a long-term goal – being able to imagine how something should function in the future and being able to stick to this goal over time – can also be connected to data-driven development. The ability to break down a problem into smaller pieces – and the patience of working diligently at one piece at a time and not giving up – does require some grit. Many programmers I interviewed talked about patience in relation to grit. I asked Sam in Silicon Valley what he meant with patience when he mentioned it in connection to debugging. He told me that it took time, effort, and patience to debug: "Sometimes you run a job and it takes one or two days to find the small little error that ruined the whole model". Similarly, Daniel at the Daily underlined how often code and programs failed: "You fail, fail, and fail again until you succeed, and then you

4. In psychology, grit is considered important for succeeding in something, alongside external motivations (such as money or status) and inner motivation (performing an activity because it feels good in itself). Grit as a factor of success has been highlighted by psychologist Angela Duckworth (2016) and is defined as the power of passion and perseverance. As a teacher, she realised that IQ wasn't the only thing characterising successful students, but also grit. Duckworth and her colleagues (2007) conclude that individuals high in grit are able to maintain determination and motivation over long periods despite failure and adversity. Hence, according to them, grit is a better predictor of success than IQ, talent, or genes.

take the next problem and you will fail, fail, and fail again, and so on". He thus highlights the need for patience and the necessity of preparing oneself for code and software not immediately functioning as desired, as most often they don't. Rosenberg's (2008) account of how a development team worked for four years in their quest for "transcendent software" is an example of this quality of sticking to long-term goals, and not giving up, even after countless setbacks. Also, Brooks (1975) emphasises that if something fails, admit it frankly, but above all, find another way to tackle the problem.

Grit is thus connected to not giving up, which in turn is connected to believing in oneself (as discussed in Chapter 4).[5] The programmers I interviewed told stories of struggling with a problem, and when they succeeded, they were recognised for their efforts. "You have to believe in yourself, fight for your ideas and not give up", as Anna in Silicon Valley declared. Hence, grit is about not abandoning a problem and your vision of how to solve it, and when hitting a wall, not giving up, even in the face of people telling you that you will never succeed. This is how Steve Jobs is often described, not as having magical powers or superhuman abilities, but sheer determination and cleverness (see Fisher, 2018). It seems as if a combination of grit and genius allowed him to achieve his icon status in the culture.

Grit also requires a kind of "I-can-do-it mentality". This is exemplified by Nadja in Silicon Valley, who shared with me her story of coming as an immigrant from Lithuania to the US. She was put to what seemed to be an impossible task in terms of a deadline by the boss in the company that first hired her. She complained to him that it was impossible to achieve this within the set timeframe; however, she was told that if she didn't deliver, she would lose her job. She couldn't risk losing her job, because if she did, she would have to leave the country within 30 days, and at the time, she had two children to support. So, she didn't give up at the seemingly impossible task; she put her mind to it, worked 24-7, and made it happen. Kristina, the programming student in Berlin, also emphasised the I-can-do-it mentality. She claimed she was willing to learn, was curious, and wanted to be part of the programming community and all the "cool projects that will change the world", as she phrased it. "And I still believe, even though I'm soon 40, I can do it, I can do it… If you want

5. While grit is considered important in the programming business, there are examples when programmers actually should give up. The whole story of Elizabeth Holmes and her startup company Theranos is a good example. The HBO documentary *The Inventor: Out for Blood in Silicon Valley* (Gibney, 2019) is a painful testimony of how a general embrace of entrepreneurship and startup self-confidence, mixed with a refusal to give up on ideas, can lead to fraud, threats, and explicit deception of funders as well as the general public. Holmes claimed to be able to run all kinds of tests out of just one drop of blood in her magic machine Edison (interestingly named after the inventor who never admitted he failed). This tale of a woman, who in 2015 was celebrated as the next Steve Jobs (Stevenson, 2015), is a reminder of the importance of critical thinking, questioning, and not being swayed away by entrepreneurship and startup hallelujah.

you can do anything". I commented that she seemed to have a lot of trust in herself, and she told me how she overcame self-doubt:

> I'm also dealing with some problems with self-doubt, that I should take any job and forget about my dreams… but two of my teachers told me maybe I'm not among the best students, but they told me I'm motivated and that is important. They told me I'm over-motivated actually […], that I will be good on the market because I'm very motivated.

Kristina wasn't particularly young and therefore claimed to not be the quickest in coding or picking up coding problems. She told me she worked entire evenings and early mornings in order to keep up with her younger classmates. But she had the will and motivation, which was also acknowledged by her teachers. Interestingly, in a similar manner, middle-aged Ted in Malmö argued that grit was a quality that younger programmers had a hard time with. Younger programmers weren't as persistent as older ones, he claimed. I asked him to develop this line of reasoning, and he explained that younger programmers were more motivated by new things and embarked on the newest, latest, and coolest, what they found the most exciting at the moment. Older programmers are, according to Ted, more gritty and diligent, so there might be room for older programmers in this youth-oriented culture after all.

I have talked about tech culture being male coded. Ted, while reluctant to define grit as a male characteristic, did agree that it was more appreciated in societies in general when a man became obsessed with something and super focused on one particular thing, than if a woman would become equally obsessed. He related this to his experiences of good programmers having played computer games as kids, and not stopping until they "aced the game, found the princess, and solved the problem". He referred to the ability to play a game over and over again, becoming infatuated and learning everything about it in order to crack it, as well as finding immense pleasure in doing so, a pleasure coupled with a belief in eventual success. I asked if this is one reason programming is so male-dominated, but he argued that there were other explanations as well; for example, many women aren't as single-track minded as men, and women are often more prone to juggle many projects simultaneously. As Graham (2010: 29) argues, if a program isn't doing what it is supposed to, programmers find out where it went wrong because "you know you are going to win at the end"; women don't seem to have this same "holy calling" as men, according to Graham (2010: 76).

When having debugged for a long time, the moment when it all works is very joyful. Bart talked about the pleasure of predicting how his code would work in a particular way, and how he just loved when he was proven right. Sam described the best workday as a day when he had an "aha! moment": "You have been struggling for something for a long time and it all of the sudden comes together, and aha… we nailed it!" Or in the words of Nadja:

> You have something that works and you get an adrenaline rush… something compiles and you see outcome, or some application starts to work the way you want and you have an adrenaline rush. […] You always get an adrenaline rush when something is working.

I then asked Nadja what a bad day was like. Is it a day is when nothing is really working?

> Yes, and you debug for five days without getting anywhere… and you have a deadline or something and you are stressed…

Ensmenger (2015) labels this as an obsession with the eureka moment. Similarly, Ali Aydar, a programmer for the legendary peer-to-peer file-sharing program Napster, talks about the technology itself as a eureka moment (in Fisher, 2018: 289). Hacker Bill Gosper compares the confirmation of the correctness of his code with the feeling of becoming a father (in Levy, 1984). The pleasure of the eureka moment can be connected to grit; it's easier to be gritty if one knows that the moment it comes together is deeply fulfilling.[6] In Silicon Valley, it is common to throw a big party after reaching a milestone, with free alcohol for employees and rented inflatable castles to jump around in like overtly excited kids (Gibney, 2019).

The mind-blowing demo

An interesting ritual in tech culture is how new devices and functions are presented to us. According to tech historian Randall Stross's (2008) biography of American inventor Thomas Edison, it was because of Edison's public demonstrations that he gained his wizard status. Like magicians on stage, the presentation of technology should have a "wow factor"; the audience is expected to be blown away by what is presented. Programmer Tom Pittman, for example, is described as taking pleasure in evoking astonishment from people by telling them he had made systems perform beyond their theoretical limits (Levy, 1984). And perhaps the most iconic example of this ritual is Steve Jobs, in his blue jeans and black turtleneck in front of an enthusiastic crowd of Apple co-workers.

Having visited meetups and tech conferences around the world, I have countless examples of a brisk moderator reminding me that what I'm part of or witnessing "is just great", with the added question, "don't you just love this?"

6. As Chandra (2013) explains in his book, when he fixed code, the victory coursed through his brain and body. Every time a piece of code worked, it gave him a jolt of joy, and when it didn't, his world fell away; body and time vanished. "Three or five hours later, when the pieces of the problem came together just so and clicked into a solution, I surfed a swelling wave of endorphins" (Chandra, 2013: 19). At those times, when everything fell into place and he found harmonies and symmetries in his code, he became a mystic, a seer of some sort. Chandra ends his book by referring to this as a reward of both curiosity and patience, a double presence of art and logic, which he also connects to ancient Sanskrit teachings.

During an Apple product presentation in June 2014, journalist Dan Frommer (2014) counted more than 50 outbursts of laughter and more than 100 stops for applause from a crowd of over 5,000 people during the 117-minute presentation. Indeed, the audience has a role to play. The mind-blowing demo is closely related to entrepreneurship and startup culture, with its belief in changing the future for the better through the devices programmers develop.

I must admit I have been blown away many times while doing research for this book, mostly by the sheer scale of things and the amount of money invested in, for example, art installations at SXSW. The exhibit hall of SXSW2019 is another example where I was overwhelmed by the multitude of virtual and augmented reality gadgets to use: robots, lovebots, sushi-making machines, and even "living" animals to pet. One art installation that captured my attention was labelled HASH2ASH (by a group called Pangenerator) in which my selfie – which I tagged and uploaded online – was displayed with black and white gravel on a 1x1 meter screen, before disintegrating into a pile of gravel (see Figure 6.1). This was supposed to symbolise that data on myself will eventually fade away – something I doubt. Still, it was a pretty cool art installation.

Figure 6.1 *Hashtag to ashes*

Source: photo by Jakob Svensson

My friend and colleague, sociologist Martin Berg, brought my attention to the fact that the starting point when demonstrating new technology is seldom the practical functionality of the device. Rather, what is highlighted is the device's

revolutionary, mind-blowing, and often magical qualities – regardless of whether people really need them or not. After watching several Apple demonstrations on YouTube, I realised Martin is right. Magic permeates these presentations (and not only when demonstrating the Apple Magic Trackpad in 2010).

The mother of all demos

The mind-blowing demonstration has a history in tech culture. It all goes back to Doug Engelbart and a demonstration he held in San Francisco in December 1968. Engelbart was, according to Fisher (2018), the first to actually build a computer. Coming from the Navy, his idea was that computers should augment rather than replace the human intellect (see also Markoff, 2015). Engelbart received funding to test different types of display selection devices, and this is when he came up with the idea for the mouse, according to himself (in Fisher, 2018). The mouse was also grounded in a perception of computer technology as interactive and for personal use, a controversial idea at the time.

Engelbart and his funder Bob Taylor wanted to demonstrate the computer and its functions, and they planned to bring in a huge screen and then have online support between San Francisco and Menlo Park to showcase, among other things, live video conferencing. Taylor was happy to pay: "spend what you need, but don't do it small" (in Fisher, 2018: 21). ARPA apparently spent USD 175,000 for that single demonstration. Engelbart and his team set up links, organised with receivers and transmitters, flew in two huge video projectors, and engaged computer pioneer and counterculture icon Stewart Brand to really put on a show. In other words, the scale of this demonstration "was unbelievable" (Fisher, 2018: 23) and in itself an example of the intermingling of entrepreneurship and hippie culture, with Brand's experimentation with psychedelic patterns projected on the background screen.

When the lights came up on 9 December 1968, Engelbart sat on the stage with a giant screen behind him and a mouse at his fingertips to show what his computer could do. The presentation demonstrated almost all the fundamental elements of modern personal computing, such as windows, hypertext, graphics, efficient navigation and command input, video conferencing, the computer mouse, word processing, and a collaborative real-time editor. According to Engelbart himself (in Fisher, 2018), this was the first time the world had ever seen a mouse, hypertext, mixed text and graphics, and real-time video conferencing. This might not seem so revolutionary today, but in 1968 it was; the audience was used to operating computers with punch cards. The thousand or so people in the audience were blown away, with one participant referring to it as an "otherworldly experience" that he couldn't "quite believe [...] was all for real" (in Fisher, 2018: 25). It seemed like magic. When the demonstration was over, Engelbart received a huge standing ovation (some parts of this demonstration can still be accessed; see MarcelVEVO, 2012).

This presentation became known as the "Mother of all Demos" (see Fisher 2018: 24). Engelbart was considered a hippie "crackpot" to a significant portion

of the computer science community prior to this demonstration (Metz, 2008: para. 4). But when he was finished, he was described as "dealing lightning with both hands" (see Markoff, 2005: Ch. 5). The Mother of all Demos set the tone for how new technology would be presented from then on, until today.

Frommer (2014) writes that one of Apple's most successful products isn't made of aluminium and glass, but words and pictures. The presentations, or keynotes, are Apple's tool for unveiling products to millions of people. It's possible to argue that this began with Apple's first commercial in 1984, directed by famous director Ridley Scott (see The Retronaut, 2014). The theme, "a new world is coming", was apparently an allegory against IBM. When the computer was demonstrated, Steve Jobs argued from the stage that there were two milestones in the history of computing, the Apple II and the PC: "Today we are introducing the third milestone product, the Macintosh" (in Fisher, 2018: 113–117). He then pulled a Macintosh out of a bag, plugged it in and walked away, and the computer started to speak:

> Hello, I'm Macintosh. It sure is great to get out of that bag. Unaccustomed as I am to public speaking, I'd like to share with you a maxim I thought of the first time I met an IBM mainframe: Never trust a computer you can't lift. (see macessentials, 2009)

Jobs (or the Macintosh) received a standing ovation, and according to Jobs himself (in Fisher, 2018), people were crying. Another of the most famous mind-blowing demos is when Steve Jobs unveiled the first iPhone on 9 January 2007. Apparently, the energy in the room was amazing, electric (see Fisher, 2018).[7] This particular style of presentation came to be known as the Stevenote. A well-known feature of the Stevenote is to feign some concluding remarks, but then return to the stage saying "but there's one more thing", in which the highlight of the presentation is then revealed: the hot new product that millions would buy. Jobs therefore surrounded this climax with context and suspense. Stevenotes even caused substantial swings in Apple's stock price (Wikipedia 2020e; see also Rotenberg, 2014).

▼▼▼

In this chapter, I attended to the rules and rituals of programming. The logics of data-driven development, the state of flow, and the importance of grit all highlight that programming is a process that should lead to something new, otherworldly devices that are then demonstrated in mind-blowing demonstra-

7. The iPhone developers were sitting in the third row and had a drinking game going: every time someone's work was presented on the big screen, they got a sip of the vodka bottle being passed around. Eventually they got very drunk.

tions. This is when the magic is showcased and the future is revealed. Development is based on *data*, underlining its special significance in the culture. Also, to become one with the machine and believe in oneself is how programmers provide their profession with meaning, which leads me to the next chapter on values and imaginations. Tech culture values the future, and not just any future, but a better future that programmers will contribute to with their coding skills and imaginations, providing solutions to problems in creative and innovative ways. These are imaginations centred around data and its hidden patterns.

Chapter 7

Innovative & creative solutions for a better future

In this chapter, I attend to beliefs, imaginations, and values in tech culture, or the *core* of the culture, following Geert Hoftsede's (1991) analytical model. As Robin Mansell (2012) argues, technology is shaped by imaginations: social and cultural perceptions and values guiding the evolution of communication systems.

I start by addressing the value of solution creativity, which gets to the heart of what motivates programmers to spend hours in flow and not give up. Tech culture is also future oriented, and in the second section, I thus discuss programmers' fundamental belief that through technological solutions, the world can become a better place. Programmers just need to believe – not only in unicorns, but also in the possibilities of technology and progress – and embrace the skills of both science and art, as Prometheus bestowed on the people. Third, I address the values of disruption and innovation, which can be connected to the underdog position and a general anti-authoritarianism. And as with the beanbag, disruption and innovation symbolise an embrace of differences, as well as underlining the importance of flexibility, imagination, and grit. I end the chapter by discussing imaginations of data, the belief that through enough data – and the right kind of data – programmers may find hidden patterns through which almost any problem can be solved and any imagination fulfilled. And with that, the circle is closed and I'm back to the value of solution creativity.

Solution creativity

In previous chapters, I have discussed the importance of artistic skills, having grit, and finding flow in programming. Behind all this is a quest to find solutions, and not just any solutions, but elegant solutions, beautiful solutions, programmers' *own* solutions – in other words, solution creativity. Tech culture is indeed more about solutions than problems. Frederick Brooks (1975) asks rhetorically: Why is programming fun? One reason he provides is that programming has similarities with solving complex puzzles. According to Emily Chang (2018), programmer scouts in the 1960s thought that people who enjoy solving puzzles make the

best programmers. In Scott Rosenberg's (2008) book, one of the programmers also talks about how he loved to solve puzzles as a kid. Though programming is still about puzzles, dilemmas, and providing solutions to all kinds of problems, complexity is pushed to the background or dealt with later, as with the practice of data-driven development. Indeed, programmers are so eager to get to solutions that unintended consequences are often downplayed or even ignored.

Almost all the programmers I interviewed told me they were motivated by solving problems, and it did not matter if they were formulated elsewhere and not by the programmers themselves. This attitude was especially prevalent among those working freelance (obviously), but also among programmers in big tech companies. Martin at Facebook, for example, referred to himself as an applied scientist, as someone who tried to solve problems. He claimed this was the reason for his exit from academia: "Let me tell you one thing, I'm not what you would call a brilliant scientist. […] What I am is a problem solver and an engineer at heart". Nadja in Silicon Valley echoed the problem-solving sentiment when she described her perfect workday: "You get an adrenaline rush, something compiles and you see the outcome, or some application starts to work the way you want it to work". In this sense, solution creativity is connected to the pleasure of the eureka moment. When immersed in difficult problems for a long time, sometimes even for years (see Rosenberg, 2008), it is extremely rewarding for programmers when they finally solve them. This is not only about grit, but also about a solution orientation in tech culture, and to *do* things ("play hockey") rather than just talk.

I first came across the value of solution creativity when talking to Pelle in Stockholm. I asked him about his motivation for becoming a programmer, and he described the joy of solving problems. He told me that when he came up with an idea of how to break down a problem into smaller modules in order to solve it, and if the idea actually worked, then it was very satisfying. But if he encountered bugs – which programmers often do – then he needed to rethink his solution ideas. And for this, solution creativity was important, or in the words of Pelle: "It's about getting your idea to work, that your *solution creativity* works… it's this kind of creativity you're hunting for" [emphasis added]. When I asked him what he meant by solution creativity, and whether a particular interest in problem-solving is needed, he underlined that there is always more than one way to solve a problem:

> As a programmer, often you dream not so much about getting a certain job or position, but *more about how to solve a problem* in an elegant manner. […] I mean… your ideas of a new product or service are important, but of equal importance is your manner of solving problems, tackling problems, how you go about things. […] *It's not the problem that is the problem, but how to solve it*. And it can be solved in different ways. So, it's about solving

> it in a way that is elegant, easy to understand and maintain. […] The best is to find a solution to a problem that nobody else has solved, and solve it in a good way, *prove that your ideas work, and that the problem is solvable*. […] Sometimes there are programmers who *do things just to prove it can be done*, or do it in a different way, to prove they can do it. That is the challenge. [emphasis added]

Solution creativity is thus connected to solving problems in steps and the elegance of bug-free and understandable code, as discussed in previous chapters. Indeed, algorithms and automated systems are about problem-solving calculations in steps. In *Algorithms for Dummies*, algorithms are defined as solution-finding technologies, and according to the authors, finding solutions should be fast and easy (Mueller & Massaron, 2017). Kim in San Francisco, underlined that before programmers begin coding, they must think about the algorithmic design – how to tackle the problem. It thus seemed as if problems most often were given, or formulated elsewhere, and that programming was about solving them creatively, iteratively, and with elegant code.

Important to this problem-solving process, however, seems to be that the solution is new and original, something that the programmers *themselves* have come up with. Learning from others is fine, but copycats are frowned upon. I asked Lasse in Stockholm about the best way to solve a problem, and he answered with a laugh that it was *his* solution, and underlined how motivating it is to find a solution to a problem that no one else has solved:

> It's about proving that your ideas work, that you can solve the problem in *your* way, and often you just want to show that *you* can do it, that it's possible. [emphasis original]

Programmers do indeed seem to be motivated to prove that their own solutions work, but it is equally important to solve problems in a beautiful and elegant manner, and it is here there lies an opportunity for creativity. Legendary American programmer Andy Hertzfeldt asserts that there is a third set of values in Silicon Valley: "Beside financial and technical ones, there are the values of the art and the artistic, the values when you want to create something new […] then it isn't your tech skills that matters, but originality" (in Fisher, 2018: 11–12). Bart, the Dutch programmer I interviewed in Copenhagen, emphasised this:

> You can solve a problem in a million different ways. You just have to find a way that is beautiful, elegant, and simple, and hence this is a very creative process. It's not like I have to do this, and there is only one way of doing it, and now I have to just write the code down as fast as possible. No, you can do it in a million ways.

In Lund, I asked Adam whether *solving* the problem is more fun than the *actual* problem, and he agreed it was. Adam also emphasised that for many

programmers, having a background in hacker culture – to do something just because they *could* – was motivating, to *prove* they could do it. The mind-blowing demos of new technology discussed in the previous chapter are also about proving what one can do or accomplish; however, showing off cool functions are at times foregrounded at the expense of user demands and usability. Yet, solving problems and creating demands that users don't even know they have is a way of inventing the services of the future (to which I will return to in the next section).

Adam also underlined solution creativity as solving problems *differently*, as "thinking outside of the box". I saw this statement painted on the walls when walking around tech headquarters in Copenhagen as well as in Bangalore. Paul Graham (2010) argues that this is a distinctively American characteristic. He argues that Silicon Valley is placed in America – and not in France, Germany, England, or Japan – because "in those countries, people color inside the lines" (Graham, 2010: 53). I'm not sure I agree. And Martin, who is Danish but works at Facebook's headquarters in the heart of Silicon Valley, explained that he had a completely different way of thinking, which was needed in the Valley, "because we Scandinavians can think outside of the box". Working with future products, he argued his job was "all about thinking outside of the box and being creative".[1]

When interviewing Roger, a Google employee, he claimed that all the smart people working at his unit were drawn there because of their fascination with solving problems. Programmers "gravitate towards something that captivates their hearts and really hard problems that will engage their minds", as he phrased it. To solve difficult problems creatively was also good for building their reputation as programmers (see Turner, 2009).

Solution creativity is thus a value that revolves around difficult problems and their creative solutions. And at Google, it is even possible to argue this is *forced* creativity, since Google expects their employees to have new ideas and allow them to spend 20 per cent of their working time on projects of their own choosing. According to Fred Turner (2009), 50 per cent of the products Google launched in 2005 were created out of this 20 per cent time allowance. To see "your solutions materialise in a product" is indeed rewarding, as Vikas in Bangalore explained to me. Though the focus on creativity may seem like providing tech workers with a great deal of freedom, some tech employees I interviewed were also burdened by the fact that creativity was expected from them: "All of my colleagues are brilliant, really creative, and sometimes I feel I cannot keep up with the number of ideas they generate", Gabriela in Sao Paolo

1. This is also highlighted in popular depictions of tech culture, such as in the 2013 comedy *The Internship*, in which characters played by actors Vince Vaughn and Owen Wilson, as recently laid-off salesmen, attempt to compete with much younger and more technically skilled applicants for a job at Google (Levy, 2013). It's their *difference* – their ability to approach tech problems differently – that eventually wins them the internship.

told me. I asked her if she felt pressure to come up with new ideas all the time. While not directly confirming this, she did hint that her creativity was something that would be discussed in her performance review.

The belief in technology to solve problems creatively and elegantly extended beyond those skilled in coding. For example, when interviewing Anna in Oakland, she told me that she didn't have any particular background in technology but was interested in "how we can solve problems, what solutions we can get at with technology". What got her interested in tech was "curiosity about systems, how they are built and what they are doing". Hence, solution creativity is also about focusing on the big problems that contemporary data societies face, which resonates in Facebook CEO Mark Zuckerberg's (2012) "focus on impact" as a core value in The Hacker Way:

> If we want to have the biggest impact, the best way to do this is to make sure we always focus on solving the most important problems. It sounds simple, but we think most companies do this poorly and waste a lot of time. We expect everyone at Facebook to be good at finding the biggest problems to work on.

From a critical perspective, Evgeny Morozov (2013) complains that the desire to solve all kinds of problems and the will to continuously improve is shortsighted, as it recasts complex situations as neatly defined tech problems with computable solutions. He has written about this as a "folly of technological solutionism" in his book with the telling title *To Save Everything, Click Here*. Solution creativity can also be connected to another prevalent theme in the culture, namely, the future. The programmers I interviewed weren't talking about *any* kind of solutions or *any* kind of creativity; their work is about finding solutions for a *better future*. This leads me to the next value: progress, a belief that things will get better.

Progress, the belief in a better future

During my journey into tech culture, I was continuously reminded of the future; for example, when walking around Electronic City in Bangalore, one of India's largest IT hubs, I saw company signs with slogans such as "in the service of a better future" and "manufacturing the desires of tomorrow". At the Computer History Museum in Mountain View, staff walked around in yellow t-shirts with "Predict the Future" printed in bold letters over their chests. The focus on the future was also explicit when visiting tech headquarters in Scandinavia, appearing in corporate slogans and encouraging shout-outs like "invent the future", "the future is here", and so on. Mark Andrejevic (2020) talks about this as a "temporality of the future present". Indeed, programming operates

on the assumption that the present is nothing more than the raw material from which to construct a better future.

To be in the future implies not looking back or being nostalgic about the past. When interviewing Nadja, she talked admiringly about a man who flew to Russia in the 1990s to install networks: the Glasnest (alluding to "Glasnost", Gorbachev's political slogan for increased government transparency in the Soviet Union). "No one sponsored him, he had no donor, it was done on his own initiative", Nadja commented. This man apparently believed that he could change the Soviet Union for the better by providing computer networks to the people. This story prompted me to ask Nadja if she missed those more philanthropic projects, and she answered, rather tellingly, that she doesn't miss anything because "we are in the future now". Indeed, there is no room for regret in tech; programmers must keep moving forward, moving fast towards the future and dealing with problems later. In this way, the importance of speed – to deliver quickly rather than bug-free – can be connected to tech culture's future orientation. As engineer and entrepreneur Tony Fadell phrases it, "today is the slowest day society will ever move, tomorrow will get faster and change will come around faster and faster" (in Fisher, 2018: 422). The practice of data-driven development is also connected to trying out solutions first and thinking about potential consequences later.

The epicentre of tech development for the future is Silicon Valley – perhaps not so surprising. In his closing chapter, Adam Fisher (2018) underlines that Silicon Valley is the place where the future is made. And the best way to predict the future, according to programmer Alan Kay, is to *invent* it, (in Fisher, 2018). A common characteristic of people who work in the Bay Area is to believe that they are at the centre of the world, and that a majority of what is to come in the future is founded there. According to John Markoff (2015: 317), the ethos in Silicon Valley is to have an impact and that the "future is just around the corner". One of the interviewees in Charles Darrah's (2008) article claims that Silicon Valley isn't about the here and the now, but where society is going in the future. Darrah thus argues that working in Silicon Valley involves more than programming skills; it situates people in a community marked by values around time and judgements about a good life. In this way, Silicon Valley constructs itself as the centre of a larger progressive movement, a link to the future in which digital technologies imply a better quality of life for all. No wonder Darrah's article on tech workers in Silicon Valley is titled "Techno-Missionaries Doing Good at the Center" (see also Chapter 4). Some critical scholars, like Morozov (2013: 5), lament this will to improve and consider Silicon Valley as "improving about everything under the sun". As former Google CEO Eric Schmidt allegedly said, "technology is not really about hardware and software any more, it's really about the mining and use of this enormous data to make the world a better place" (in Morozov, 2013: vii). Similarly, the hackers in Levy

(1984: 37) also wanted to improve the world through the computer: "If everyone could interact with computers with the same innocent, productive, creative impulse that hackers did, *the Hacker Ethic* might spread through society like a benevolent ripple and computers would indeed change the world for the better".

Young Nilesh in Bangalore explained to me that through technology, the future can be really positive for everyone. Technological improvements and solutions were supposed to be universal and not just limited to the upper classes (see Darrah, 2008). Anna in Oakland connected her work with her interest in social change issues in a similar way: "I really do think that technology has the power to be a great equaliser". She relayed her story as a non-binary kid, how she didn't see herself reflected in anyone, and she contemplated how her life would have been different if she would have grown up with the Internet and the opportunity to connect with like-minded people. She further reflected on her son, currently working on a school essay on the Israel-Palestine conflict, and how he, via the Internet, could talk to people "on the ground". She used this as an argument for how technology can bring people together and provide a better and more connected future.

The future is painted in bright colours in tech. Brooks (1975: 14) writes that all programmers are optimists, and "this modern sorcery especially attracts those who believe in happy endings and fairy god-mothers". This optimistic belief in the future can be summarised in one word: *progress*. Markoff (2015), for example, describes Silicon Valley as full of optimist technologists who believe in technical progress. Progress is about embracing change, since things tend to become obsolete rather quickly. Almost 50 years ago, Brooks (1975: 115) argued that "there is nothing in this world constant but inconstancy". He continued by asserting that people should "accept the fact of change as a way of life, rather than an untoward and annoying exception" (Brooks, 1975: 117). Hence, programmers must plan for change. Apparently, this is what puts Silicon Valley at the forefront, as it "has the least resistance to new ideas", according to writer and editor Kevin Kelly (in Fisher, 2018: 432). And when programmers believe that this change is inherently positive, it becomes progress.

The task of many tech development teams is thus to experiment for the future, situating the practice of tinkering in a culture of entrepreneurship. At the Daily, experimenting with a news-ranking algorithm was motivated by a quest to develop "the future of journalism" and what a "modern newsroom could look like", as in-house programmers at the newspaper told me. Already at the famous Xerox PARC, the employees considered their task to be developing "the office of the future", an office which should be computerised. Decades later at SXSW2019, I attended a seminar on the blockchain technique, a technique which is claimed to "change the world", and therefore, this is the "most exciting time to be alive ever". At the opening keynote, the audience was also encouraged to "do the impossible" and "invent the future".

Thus, the belief in progress can also be linked to technological solutionism. Morozov (2013) approaches tech solutionism as an end-of-history argument, where the Internet is conceived of as the finest, final, and ultimate technology of humanity. Technological solutionism is the conviction that people today are living through unique and revolutionary times, and this opportunity must be seized in order to ameliorate the world and the human condition. Arguably, entrepreneurship and startup influences accentuate this faith in technological innovation to solve the big problems the globe is facing today – such as power generation, transport, food, manufacturing, and building – as a way to prevent rampant climate change. Quite a few of the programmers I interviewed – both in Scandinavia and Silicon Valley – had at some point worked on climate change projects, work they were eager to talk about. Roger at Google spoke about his endeavours in Africa and Estonia, Niklas in Norrköping worked with an energy consumption project, and Hans in Malmö had been involved in using tech to promote local farming (see also Chapter 4).

According to Richard Barbook and Andy Cameron (1996), it is the belief in a better future through technology that makes it possible for tech workers to simultaneously embrace market capitalism and hippie ideas of hidden networks that bind not only individuals, but also all living beings, together as one. Today, programmers are a privileged part of the labour force, heirs to radical ideas of the left and media activism, simultaneously reflecting market economy and the freedom of hippie artisanship. This bizarre hybrid, discussed previously as the Californian Ideology (see Chapter 4), is only made possible through a profound faith in the emancipatory potential of technology.

Disruption & innovation

Coupling the value of progress and the embrace of change with the entrepreneurial side of tech culture results in disruption; this is *similar* to solution creativity, but not necessarily the same. Sometimes, disruption is about inventing the services of the future by creating demands that we in front of the screen didn't even know we had. So-called creative disruption refers to radical change brought about by overturning existing conventions. Disruption can thus be connected to the general anti-authoritarianism and the embrace of the underdog position in tech culture.

Even though Silicon Valley is seen as the centre of technological innovation, tech workers there still have a conception of themselves and their products as radically different from the rest of the world. One example is the mind-blowing demonstrations of new products referred to in the previous chapter. These products would *fundamentally change* how people, for example, listen to music, take pictures, or navigate traffic. How AirBNB disrupted the hotel market, or how Uber disrupted the taxi market, are often-cited success stories. In business

theory, disruptive innovation is what creates a new market which eventually disrupts an existing market (see Bower & Christensen, 1995). There is even a term for when a whole business model is wiped away with the introduction of a new service: "Netflixed", referring to how the streaming platform erased the whole rental movie business (Harrington, 2013). Indeed, as Chang (2018) laments, at Google, hubris is rewarded, and there is a preference for so-called blue-sky projects, such as driverless cars and curing death. Disruption can in this way also be connected to a belief in oneself as a programmer, grit, and not giving up on one's ideas. Conceiving of disruption as something positive is a consequence of considering oneself as capable of doing what others have not even thought of – someone who literally "invents the future".

The question is whether it is still possible to talk about disruption, or whether it has become more of a myth. In Silicon Valley, many startups are today quickly bought up by tech giants whose dominance is *not* disrupted, as they keep getting bigger and richer. In his critical book, *The Black Box Society*, Frank Pasquale (2015) cites claims made in 2009 that no one would care about Google in five years, based on the premise that something younger and newer would wipe it out. In this sense, disruption is as much a value as it is an imagination – digital economy by nature being open to being disrupted. In management, there is even a book with the title *What would Google do?* (Jarvis, 2009). The answer Pasquale (2015: 99) sceptically gives to Jarvis's question is that Google would "use their data to outflank competitors and extract maximum profits from their customers". The same story is told in the "Amazon Empire" documentary; Amazon outcompetes company after company, and also whole industries, which leads to a monopoly for Amazon in a number of areas (Jacoby et al., 2020). I thus wonder whether these tech giants are becoming the equivalent of the evil IBM regime that they themselves so despised in the past, "thriving on monopolistic manipulation of the market place" (Levy, 1984: 186).

With regard to financial incentives and creative disruption, there was a difference between the freelance programmers I interviewed and programmers working in big tech. Some of my participants mentioned the quest for disruption and focus on money-making as a major reason for them going freelance. They juxtaposed their rejection of a money incentive with their motivation to help people, as Python in Berlin explained:

> It got a bit tedious because it was about the money […] seeing all pre-sales meeting and how they push things that don't really fit the customer well… just for the money, I didn't like that.

With the increasing importance of entrepreneurship and how the previous underdogs of Silicon Valley have become dominant, there is a development in tech culture towards commercialisation, with values connected to market dominance and profit-making. Apparently, "domination" was a mantra at Facebook during

its early days, something screamed out by Zuckerberg and others when on their way to actually becoming dominant (Fisher, 2018). Pushing for increased profits is part of being an entrepreneur, as Narendra in Chennai explained to me. But disruption is not only about making money through overturning an existing market; it is also about fun and doing things because they are possible. As Ted in Malmö phrased it, "when I started this company, I lowered my salary by 50 per cent, but I have more fun today". Similarly, Roger at Google underlined his motivation as "trying to have a positive impact with technology", concluding that his purpose had always been superior to profit. Fisher (2018: xv) claims that Silicon Valley was never about profit, but about "resistance, heroism and struggle"; however, at the same time, in the chapter on Google, readers learn how happy Brin and Page were when they started making money. Indeed, there is an indecisive association with market capitalism and profit in tech culture. But regardless of whether programmers are motivated by money, it seems that profit is good proof of success, that one's "ideas and solutions are correct and valuable", as Narendra in Chennai asserted.

Disruption is apparent throughout the stories in Fisher's book, but to disrupt, programmers must have a bit of craziness and boldness: to be unafraid, "to stay hungry, to stay foolish", as Stewart Brand phrased it (in Fisher, 2018: dedication). Phrases reflecting the tendencies to think big and outside of the box are prolific in Fisher's interviews; for instance, "this is going to change everything" and "this is amazing", phrases that also echo in the tech conferences I attended. John Markoff even claims that Silicon Valley is about commercialising and disrupting normal businesses (in Fisher, 2018). There seems to be an aversion against the taken-for-granted and traditional ways of doing business.

Disruption is therefore connected to innovation, which is treated as inherently good regardless of the consequences. I lost count of the number of times I saw the slogan "innovate or die" during my journey into tech culture. And, according to Anna in Oakland, the real currency in tech is innovation:

> People that are *creating something new, different, interesting… disruptors definitely*, but it's not disruption that is the thing, it's the innovation. It's seeing the same thing that we all see and seeing something different in it, different possibilities. This is really the currency of Silicon Valley. It doesn't matter what education you have, what your gender is, it doesn't matter. [...] I mean, how many times have I gotten frustrated at taxi drivers, and then they created Uber, I find that intriguing. […] So, if you take away this thing with money and getting access to money, *the currency in northern California is ideas, innovative ways to solve problems*, problems people have been trying to solve for a long time. [emphasis added]

That people seek a grander meaning to their jobs is certainly not new. What is perhaps distinctive in the accounts from the programmers I interviewed is

that their work – their products and their services – are supposed to transform society to the extent that they might disrupt a whole business and previous way of doing things, and hence, invent the future. Or, in the words of Pelle in Stockholm, to prove that it is possible to do something is important, he was right and the rest of the world was wrong. There is some kind of heartless masculinity lurking in the background here, with techies worshipping "the God of creative destruction" and praying at "the altar of innovation", witnessing fine businesses coming and going without shedding a tear, as Morozov (2013: 21) laments.

The embrace of boldness, craziness, and rebellion against established markets and companies also resonates in tech culture's youth orientation. The value of being creative and solving problems rather than having experiences and routine can be connected to ageism. Nadja, who is in her 50s, put this quite bluntly: "experience isn't as valued as being creative". The young are expected to be more creative, more future oriented, and less managerial. Brooks (1975) connects this expectation to programming being populated by the young, and that young people also tend to be more optimistic. So-called pioneers are unafraid and innovative, characteristics connected to youth. The essence of pioneering, Levy (1984) explains, is about doing something brand new, discovery, having the courage and willingness to take risks, and trying to make the impossible possible.

In data & patterns we believe

The imagination that the future will be better and that all problems can be solved with programming is built upon a profound belief in data, that with enough data (and the right kind of data) nothing is impossible. Data (or information) becomes the link between the world of mere mortals and the magical universe of tech, as algorithms and automated systems act upon people in the "real world". This is a legacy from the early hippies and their use of psychedelics: the belief that information (data) connects users to a higher universe as well as all living (and dead) beings on the planet. This connection opens up the possibility that the seemingly impossible could happen here, today, and that all the problems humans face can be solved.

That data is just out there – free to collect, extract, or harvest – was a common theme in my interviews. Hans in Malmö, for example, talked about "collecting as much as possible of the data out there" for testing the software he is working on. Douglas in Stockholm referred to data as something "to harvest from users", as if humans were crops that, if carefully cared for, will produce plenty of useful data. Thiago in Sao Paulo stated that data are the building blocks of societies, and that humans and nature "are just data". Adam in Lund argued that if he had enough data, he could pretty much do anything and solve any problem. Similarly, Pelle in Stockholm argued that the more data he had, the

better: as long as the data is correct, the algorithm or automated system will work its magic. Sunil, the young Indian programmer in Berlin working on automated cars, similarly claimed that with the increasing amount of data collected, it would be possible not only to predict behaviour, but to understand consciousness – that is, the mind – and thus be able to engineer an intelligence which would have both free will and ethics.

Given these accounts, it is not surprising to hear claims that data is perceived of as the new oil, the raw material of the twenty-first century, and the fuel of the information economy that will continue growth and well-being (The Economist, 2017; see also Pasquale, 2015). In the "Amazon Empire" documentary (Jacoby et al., 2020), it is told that it was the realisation of the value of data that made Bezos one of the richest people in the world. It is all about data, and the one with access to the data rules, hence Amazon's quest to collect as much data as possible, not least through its Kindle (see, e.g., Paul, 2020).

Shosanna Zuboff (2019), in her theory of Surveillance Capitalism, refers to human experience being claimed as raw material for datafication as *behavioural surplus*, as well as *rendition*. Pasquale (2015), in *The Black Box Society*, points at a dual meaning of the black box as both a recording device (like in airplane crash investigations) and a system that is hidden from its users. Companies assemble data on customers who seldom know what it is used for. John Durham Peters (2015) argues in his chapter on Google that the company portrays itself as a search service, when it is in fact in the data-mining business. Google presents itself as giving away information when in fact it is *taking* information from users. Mentioning this in my talk with Roger at Google, he was quick to emphasise the pro bono projects his employer was involved in. And Martin at Facebook even became angry at my question and claimed that his employer was unfairly portrayed in the media. He then spent the next ten minutes underlining all the benevolent and noncommercial projects Facebook was involved in.

The belief in data is connected to its potential to solve all kinds of problems, tech solutionism, and that "given optimal personalization and optimal data points", so-called Big Data "will plan for us an optimal life" at no cost (Pasquale, 2015: 19). But for this to be possible, data must also be imagined as the essence of basically everything (see Svensson & Poveda Guillén, 2020). Such formulations are present already in Norbert Wiener's (1948) theory of Cybernetics, and in 1976, Joseph Weizenbaum complained about the belief that essential aspects of a person are information which may be recorded and then entered into a computer. In a more recent account, sociologist Anton Törnberg (2019) refers to the perception of data as natural and objective traces of social processes, reflecting a belief in a true and untouched social reality. As long as the dataset is big enough, all answers are believed to be hidden in there somewhere. Journalist Kenneth Cukier and Internet governance and regulation researcher Viktor Mayer-Schoenberger (2013: 28) thus define the belief in Big Data as the

"idea that we can learn from a large body of information things that we could not comprehend when we used smaller amounts".

This almost religious belief in data is sometimes discussed as data religion or dataism. Dataism promises humans all the traditional religious prizes – happiness, peace, prosperity, and even eternal life – but here on Earth, with the help of data-processing technologies, instead of in Heaven after death (see Harari, 2015). Dataism is the belief that the entire universe consists of data flows, organisms are algorithms, and humanity's cosmic vocation is to create an all-encompassing data-processing system with which humans will eventually merge.[2]

So, what is it in data that is believed to hold the key to a higher truth? Coupled with the imagination of data is a belief in patterns hidden in large datasets, which powerful algorithms can unveil. At the Daily, programmers worked with the aim of keeping readers on their web page and also getting them to sign up for a subscription. The developers were looking for patterns so they would know if someone was contemplating paying for the news. If they knew, the Daily could target an ad to them, or place specific content behind paywalls in order to give them an incentive to become paying subscribers. It was important not to annoy their readers, Per explained, "but to know enough about them to know when it would be the right time to offer a subscription deal". Hence, the Daily had a whole data analytics team working with their collected user data.

On a larger scale, the reason Google burns through petabytes of data per day is because of this belief in finding patterns in which there might lie a business opportunity (see Peters, 2015). Predictive policing, for example, is also based in this belief in patterns, that crime inclinations are hidden in correlations between data categories found in people's behavioural surplus. Data about users are fed into databases and assembled into profiles and patterns. With more data, more intricate patterns are supposed to be unveiled. Hence, the data is not acting or making decisions per se, but data is analysed algorithmically into patterns, scores, and categories, as highlighted by John Cheney-Lippold (2017) in his book with the apt title, *We Are Data*.

The belief in patterns also has a bearing on scholarly practices. An AI with enough data to correlate data points will apparently render us academics obsolete (see Steadman, 2013). With a large enough volume, data will speak for itself and people won't need theories: there will be enough data and computers powerful enough to find patterns and hypotheses on their own (Anderson, 2008); mining large datasets for patterns will enable revealing effects without experimentation (Prensky, 2009); exposing patterns and relationships whose existence was not known before will be easy and straightforward (Dyche,

2. In this way, Harari's definition of Dataism as a *religion* has similarities to van Dijck's (2014) definition of Dataism as an *ideology*, referring to the objective quantification and potential for tracking of all kinds of human behaviour and sociality through online media technologies.

2012); and finding correlations that provide a full resolution of the world will be clear (Steadman, 2013), freed from human bias and framing, transcending context, and thus inherently truthful (Kitchin, 2014). The scientist's role shifts from being proactive (suggesting theories) to reactive, collecting data first and letting the algorithms ask the questions (Croll, 2012). As Törnberg (2019) rhetorically asks, what is the point with theory when advanced technology can seek patterns and discover laws and regularities?

Patterns need big numbers and thus mostly work with Big Data. It is by collecting *enough* data that not only are the past and present mapped, but the future is predicted (Andrejevic, 2020). And the more the coin is flipped, the more the result will converge upon the precalculated probability (Steiner, 2012). Indeed, patterns are about prediction, since adding an obsession with the future to the belief in patterns results in predictions. Wiener (1948) was already 70 years ago occupied with the ability to predict from information, and even though complex and varied, in there somewhere should be trends and patterns. The fascination with prediction goes back to philosopher Gottfried Wilhelm Leibniz, who thought humans are programmed to behave in a certain manner (according to Steiner, 2012). By knowing what humans say, it is possible to know who they are, and thus they become easier to predict. Törnberg (2019) labels this digital empiricism or predicative positivism, referring to the use of API-based technologies to inductively seek patterns within so-called social physics and computational social science.

There are problems with this belief in patterns and their supposed predictive qualities. There is even a condition, Apohenia, which involves the tendency to mistakenly perceive connections and meaning between unrelated things (Stöppler, 2021). In academia, Geoffrey Bowker (2013) argues that computers may have data, but that not everything in the world is given. Data is always a *sample*; even Big Data is only stand-ins for phenomena of theoretical or practical importance. It is believed that if enough data is collected and processed, then everything can make sense and the world can be understood, and thus also *engineered*. However, as even Wiener (1948) wrote, predictions are always simplifications, and as Morozov (2013) argues, simplifications are political. And who is behind these simplifications? Often the programmers themselves. By believing that data and hidden patterns can predict and invent the future, programmers make themselves into mediums who stand between the physical world and the magical world of data. This provides a nice bridge to the next chapter.

▼▼▼

In this chapter, I approached the core of tech culture through programmers' values, beliefs, and imaginations. As stated in the beginning of this book, these have a bearing on users in front of the screen, and are thus tightly connected

to the shifting power configurations in contemporary data societies. Given the increasing importance of digital technology, what programmers present as creative solutions are also about the organisation of existence, judgements about a good life, and what a better future should look like. I return to this power of imagination (and the need to democratise it) in the concluding chapter. In the next chapter, I engage more profoundly with the many signs and symbols of magic in the culture, starting with the programmers themselves. Following the metaphor of magic, I suggest that programmers can be understood as wizards. Rather than dark magicians, wizards are more benevolent and have a more helping attitude than pranking hackers.

Chapter 8

Wizards of the web

What stood out during my journey into tech culture were the continuous references to magic; *anything* is, it seemed, possible, as long as people allow themselves to believe. Like magicians on stage, programmers perform tricks with their skills in more or less mind-blowing demonstrations of new technology. They predict the future, controlling and disrupting *this* world with their knowledge of code and mastery of programming languages. Like wizards, they imagine the impossible and bring out the magic believed to be harboured in computers.

This chapter begins at some tech conferences, where I found myself among people projected as cyborg mediums – and sometimes even gods. Differing from religion, magic in tech culture refers to the possibility of changing circumstances through manipulation: magic through code. By writing code, programmers have access to something secret and fantastical, and the tool programmers use for making magic is programming languages. I continue the chapter by introducing one of the main conclusions of this book, namely that the people behind data, algorithms, and automated systems could be described as wizards. With the term wizard, I do not mean someone engaged in black or dark magic, but someone who wants to help others and contribute to a better future. With the increasing importance of entrepreneurship and the value of progress, ambivalent and secretive figures (such as hackers) have given way to more benevolent wizards, who help those of us in front of the screens with our issues as well as the larger, global problems. In the next and final chapter, I summarise my findings before presenting the concluding outlook.

Among cyborg mediums, magicians, & programmer gods

After having followed a software team for three years, Scott Rosenberg (2008) concludes that the symbols and metaphors in tech culture have a layer of magic and mythopoetic heroism to them. Technology is used to transcend the boundaries between the physical and the virtual world and perform miracles, helping users in front of the screens by first imagining the impossible and then making it happen through mastering programming languages. The programmer

stands between the frontend – the elegant, intuitive, and versatile solution – and the backend, which is supposed to be invisible, efficient, and rock-solid code and data. The programmer thus becomes like a medium, someone with access to another dimension but also with a foot in the physical dimension in which most people find themselves.

Bart, the Dutch programmer in Copenhagen, stated that he "would like to develop code that actually does something in the real world", indicating that code comes from another dimension. Bart exemplified this:

> What software can do in the *real world*, especially when you connect it to devices – like this device [pointing to his Google watch] – or have a sensor or something; that is awesome, incredible. It is just so cool that you can just write some text on an editor and then *press a button and then… bam! something that actually does something in the real world*. I am still very impressed by that *magic*. And it is still magic to me. I know how it works, but it is still *magic* to me, it fascinates me. [emphasis added]

Magic at Öredev and React Day conferences

The first day I set foot in a tech conference, at Öredev2018, I instantly noticed a plethora of magical symbols and labels, such as unicorns, sagas, wizards, and prophets. I also noticed how magic was mentioned throughout many of the talks. For example, Elsa Sortiriadis – who described herself as a synthetic biologist, bio-futurist, and science fiction writer, stated in her keynote that we (that is, the programming community) "are the co-creators of the future. Make magic!" I wrote in my research diary: magic and technology, are these people the magicians of this time? There is indeed something about these (mostly) men with their beards, loose T-shirts, and long hair – they even look like magicians. But instead of the usual props accompanying the fairy-tale magicians (like wands and triangular hats), these modern magicians use tech. One of the other keynote speakers, Teemu Arina – who described himself as a digital transformist, optimal human performance specialist, and biohacker – definitely looked like a magician with all the visible tech connected to his body. To me, it seemed as if wearable devices were physical markers of the bearers' magical skills, being connected to both the physical world – where most of us mere mortals find ourselves – but also to another virtual dimension of ones and zeroes.

Roger at Google nicely illustrated how the shortcomings of the human world can be transcended through technology. He argued that through technology, and through understanding how the brain works, it is possible to "detach yourself from humanity and some of the shortcomings of being human". When re-listening to the recording of this interview, images of shamans getting into a

trance through repeating mantras or through some other technique to detach themselves from the physical world arose in my mind.[1] Computers indeed open the door to another world, as exemplified in the many virtual reality (VR) and alternate reality (AR) demo stations which were set up in the open space of the Öredev2018 conference venue. Attendants were eager to try them out. By the looks on their faces, they seemed amazed by these applications and what they could do (the mind-blowing demonstration being a significant practice in the culture, as attended to in Chapter 5).

At the Öredev2018 conference, I attended a talk by Chris Dancy titled "History of the Occult and Technology – Downloading Paganism". In the presentation, he stated that any sufficiently advanced technology is indistinguishable from magic (citing science fiction writer Arthur C. Clarke) and that magic provides a way to talk about things that are hard to understand. He went on to state that flying carpets are magic, but when it is possible to buy one, it becomes technology. To further link technology with the magical, he argued that wireless charging is another word for energy projection, and voice interface can be compared to casting spells. Peter Nagy and colleagues (2020) talk about time management technologies with a similar vocabulary, as magic bullets that help users become more organised and efficient, and how these are sold as time-saving miracle devices. Communication scholar Paul Frosh (2019) describes tagging pictures on Facebook as technological feats of magic, connecting names to mediated bodies (images) with the aim of animating digital social networks. Frosh connects this to the ancient magical power of incantation: the use of words to perform actions at a distance. The articulation of a name makes images of the named materialise instantly before others.[2]

I managed to catch Chris Dancy after his talk in Malmö. "I see myself as a magician", he exclaimed, and continued to argue that technology could be used for good things, so "don't unplug", which is also the title of his book. At the conference, he had set up a booth called "Phone Palmistry – free reading". Like a fortune teller at a village fair, Chris told my past, present, and future, but instead of looking at my hand, he looked at my smartphone. Is it an iPhone or Android (who am I)? Do I use a phone case or have a screen protector (do I care, is safety important for me)? Is the battery percentage turned on, is the settings icon on my home screen (how do I see myself, my life energy)? Do I have a different lock screen than home screen (how creative am I)? Is my phone on airplane mode, are anti-notifications activated (how overwhelmed with life

1. Another technique would be to use psychedelics, as the hippies in the countercultural movement did (see Chapter 3).
2. Online, it is also possible to make people *disappear*. I tend to block annoying guys on the online dating app I use, and like magic, they disappear from my feed and I don't have to deal with them anymore. I similarly unfriend my ex-boyfriend on Facebook, as well as "taking a pause for 30 days" from some of our mutual acquaintances. At least, I attempted to "cast a spell" in order to not be reminded of my failed relationship.

am I)? Is there a photo of someone on my home screen, what apps cover their face, mouth, heart, or stomach (how do I feel about this person)?

The tagline of the Öredev2018 conference was "Deus ex machina", which is when a seemingly unsolvable problem in a story is suddenly and abruptly resolved by an unexpected and seemingly unlikely occurrence. The term was coined in ancient Greek theatre, where actors playing gods were brought onto stage using a machine, an image that is fitting both for how programmers are conceived of today as well as how they conceive of themselves, able to solve seemingly impossible problems in creative and mind-blowing ways. Within hacking, the idea that the computer is "out of this world" is also entertained, as one of the hackers in Steven Levy's (1984: 286) book states: "you are talking about deus *ex* machina, well we are talking about deus *in* machina [emphasis added]".

Many have made links between digital media and religion. John Durham Peters (2015), for example, has a whole chapter titled "Google as God". One of Google's founders apparently said that "the perfect search engine would be the mind of God" (Peters, 2015: 333). Google – and all user searches they have rendered into data and stored in so-called clouds[3] – evokes ancient ideas of a heavenly record containing everything ever said and done. It becomes like the "*Book of Life*", in which every deed is recorded for Judgement Day. But the all-knowing search engine also becomes like a priestly class that discerns the universe, renders order out of chaos, answers questions, and provides what users seek. Similarly, media and literature scholar Ian Bogost (2015) claims that in contemporary data societies, people find themselves in a *computational theocracy*, with algorithms and automated systems replacing gods. Levy's (1984) hackers indeed had an almost mythical belief that God is *in* the machine, a good force that they could set free with their hacking. But this belief in the positive impact of the computer goes even further back. Ada Byron Lovelace argued in the nineteenth century that her "Analytical Engine" could tabulate "any function whatever", and in the 1930s, Alan Turing predicted a machine that could "compute any function or any sequence" (in Chandra, 2013: 219–220). Through their access to data and by their understanding of how code and algorithms work, programmers are entrusted to decipher and tame the god in the machine.

In some of these accounts, it is the programmers themselves who become gods. This is, for example, how early AI (artificial intelligence) sceptic Joseph Weizenbaum (interviewed by Wendt, in Weizenbaum & Wendt, 2015) understands the Prometheus myth, not only as rebelling against the will of the gods to help humans, but by this act, also claiming to *be* a god. Fred Turner

3. Still, data is not in the cloud, but in huge data centres, transported in cables and fibers via tele-towers, and producing a lot of e-waste parks, as scholars of media infrastructures have shown (see, e.g., Parks & Staroleski, 2015).

(2006) also describes early computer scientists as gods, designing their world and channelling their disembodied energies through technology. As computer pioneer Stewart Brand famously declared: "We are as Gods and we might as well get good at it" (in Turner, 2006: 261). Philosopher Nick Bostrom's (2014) conception of programmers is similar in his book about superintelligence, where he claims that if evolution can create intelligence, so can an intelligent programmer.[4] This is an interesting thought, perceiving programmers as authors of something divine, something also apparent in my empirical material. Hans, the freelance programmer in Malmö, for example, exclaimed "there is nothing like writing code because it feels like pure creation. You have an idea of how a thing should work, and then you sit in front of a computer and make that a reality".

In the literature, Frederick Brooks (1975) compares programmers' joy of creation with God's delight in making things. And one of the hackers in Levy's (1984: 240) book described the feeling he got after hacking as "something of the satisfaction that God must have felt when He created the world". Furthermore, "The computer is a magic box, it's a tool, it's an art form […] the ultimate martial art […] an act of creation, it's where Every man can be a God" (Levy, 1984: 189).

While sharing some similarities, magic and religion are not exactly the same. While religion *requires* people to change their lives and *obey* a higher authority, magic underlines *manipulation* as a way to change the lives of others and the world around them (Fortea, 2014). Whereas religion requires faith, worship, and obedience, magic is the quest to control and dominate things *yourself*, through manipulation and access to something hidden, secret, and fantastical – through access to computer technology.[5] Indeed, tech culture is still, in principle, quite anti-authoritarian, given its influences from hippie, hacker, and entrepreneurial cultures.

Conceiving of computing as magic is especially apparent in hacking. Levy (1984) writes about the aura of mysticism surrounding the world of technology, and how computer science should rather be called witchcraft. Douglas Thomas (2002) writes similarly that hackers have become the "new magicians" who have mastered the machines controlling modern life. He mentions a hacker who dreamt of becoming a magician as a child, and that his interest in technology stems from this fascination with magic. To simply click and command becomes

4. Bostrom defines superintelligence as any intelligence that greatly exceeds the cognitive performance of humans in virtually all domains of interest (something mathematics scholar Norbert Wiener was onto already in 1948).

5. In the BBC documentary about the creation of the Harry Potter series (Harding & Ho, 2017), magic is described in a similar way, as powers out there that people cannot see but may harness for their own benefit. This theme is also apparent in science fiction literature. For example, in his novel *Neuromancer*, William Gibson (1984) talks about console cowboys, data jockeys who could manipulate the system towards their own gains.

the equivalent of using spells; just add some lines of code and "the program will do all the things you wanted it to do" (Rosenberg 2008: 2). The idea of complex technology "exoticizes hackers making them feel mysterious and capable of almost superhuman hacking feats" (Thomas, 2002: 154). Indeed, it seems that hackers embrace the idea of them being equipped with supranatural powers (as also exemplified by how they label things, such as "Avalon", the mystic place of King Arthur; see Rosenberg, 2008).

Magic is also apparent in the statements of Silicon Valley pioneers in Adam Fisher's (2018) interview book, with programmers claiming to do magic and create magical devices. What especially captured my attention was the company General Magic, a legendary American software company based in Silicon Valley, which spun out of Apple to create handheld gadgets. Already in 1994, General Magic produced a "smartphone" (over a decade before Apple's first iPhone was introduced in 2007). Fisher's (2018) chapter on General Magic is full of pocket crystals, pixie dust, supernatural links, and magic machines. At the Computer History Museum, I also saw a sign with the following inscription: "Those living before us would call it magic, but it's not magic it's software".

Magic through mastery of programming languages

Brooks (1975: 8) writes that for a programmer, it is important to perform perfectly, because "if one character, one pause, of the *incantation* isn't strictly in proper form, the *magic* doesn't occur [emphasis added]". However, as Brooks also remarks, human beings aren't accustomed to being perfect, and few areas of human activity actually demand it. He concludes that this requirement for perfection is the most difficult part of learning to program.

Mastery of computers goes back to hackers. As discussed in Chapter 3, hacking has been described as a space in which young boys could demonstrate mastery and autonomy vis-à-vis parental and societal authorities. Being able to access and explore another dimension, mainly White, middle-class, suburban boys could challenge the adult world. Hacking thus signals an ability to make the seemingly impossible happen through skills in computing. Thomas (2002) therefore describes hacking as a high-tech performance of wizardry, and that the most important element of hacker culture is the notion of mastery. Sherry Turkle (1995: 32) similarly writes that hackers are "passionately involved in the *mastery* of the machine itself [emphasis added]", and Levy (1984: 34) refers to hackers as determined to make the computer do what they wanted it to do: "like Aladdin's lamp, you could get it do your biddings". The mastery of hacking made computer programming "a spiritual pursuit, a magical art" (Levy, 1984: 146). To programmers, hackers are in many cases still conceived

of as skilled masters (see Steiner, 2012). The Greek word "techne" is actually often translated as "art and craft", pointing to the implicit meaning of the word, which is "master". Technology is perceived here as a combination of handicraft and skill, making and knowing (see Peters, 2015).

What hackers (and programmers) do is demonstrate mastery of a virtual dimension, not seldom as a response to a physical world in which they feel dissatisfied or inadequate. The notion of mastery was a prevalent theme in my interviews. Roger at Google described when he, as a kid, was first introduced to computers by his mum, and how he fell in love with programming because he could tell the machine what to do and it would do it: "I felt I could do whatever I wanted, with whatever outcome, if I just thought long enough about the input". Sören, the senior programmer at Microsoft Copenhagen, described his role at work as "the one who decides what our programs can do, what you can do with them and how they work".

When revisiting my interview material, I recognised that being in control was important (also resonating in the approach to tech as magical manipulation rather than religious obedience). When I asked them to look back on why they became interested in computers in the first place, the moment when programmers realised they were in control seemed defining. Pedro, a programmer I interviewed during SXSW2019, emphasised that the computer was a screen he could control, and in that way was different from all the other screens up to that point in his life: "I remember feeling like I controlled the TV". He also complained about the videogames he had as a kid: "There was only so many moves you could make, whereas with a computer, I felt like I was more of an equal participant in that relationship". Mark in San Francisco similarly referred to control when explaining why he got into programming (via computer gaming):

> I had this world that I had complete control over, it was a world I could master and I felt like it was possible for me to understand the whole thing and that there was nothing unpredictable about it.

According to J. K. Rowling, creator of the magical realm of Harry Potter, children start to believe in magic because they begin to make sense of and want to control their world (see Harding & Ho, 2017). In this way, magic is about empowerment, but also an acknowledgement that children are in a scary an unknowable world; hence, the impulse to try to control life itself and the future. Coupled with a belief in technology to solve all kinds of problems, mastering technology (and surrounding this mastery with metaphors of magic) can be perceived as an attempt to seek control when, in fact, control is unachievable.

Vikram Chandra (2013: 15) writes about programming as "a complete world […] I could discover and control". In particular, it is code and programming

languages programmers must master; to bring the magic out of computers and into the world requires knowledge of programming languages. If data in different combinations are the herbs that wizards use in their magic brews and elixirs, then programming languages are the spells.

System design engineer and English literature scholar Wendy Chun (2008) has written about this as sourcery (combining "source code" with "sorcery") and argues that the invisibility, ubiquity, and alleged power of source code lends itself nicely to an analogy with the magical. Furthermore, she claims that source code is "a medium in its full sense of the word as it channels the ghost that we imagine runs the machine" (Chun, 2008: 310). Approached in this way, source code (or programming languages) becomes a fetish, understood as something that is worshipped on account of its supposed inherent magical powers. Just like magical spells, programming languages act in this world, without a majority of users in front of the screens really understanding how.

Frank Pasquale (2015) also writes about the many different meanings of code; besides a general term for software, code can be a rule (as in code of conduct) or something that is deliberately hidden (as in a coded message). Code is simultaneously about something orderly – ones and zeroes that must be composed in a precise way to work – and something secret, fantastical, and creative.

Despite its magical qualities, Chun (2008) reminds us that code (programming languages) is written by humans, for humans, and read by humans. Perhaps it is this connection to humans and the physical world that makes programming languages and code accessible to people's imaginations. As mentioned in previous chapters, the world we inhabit has immense challenges ahead, and it is comforting to believe that the spells of programming languages can unveil patterns and predictions that will magically fix our problems in the quest for a better and brighter future (for an elaboration of this argument, see Svensson & Poveda Guillén, 2020). Data processed through programming languages would then be that magic elixir that not only binds people together as one holistic universe (as for the hippies), but also forms the foundation of all problem-solving (as for tech-solutionist entrepreneurs).

Connected to the idea of magic as mastery of computer programming is the idea of magic as giving life to the imaginings of programmers. Brooks argued already in 1975 that beyond craftsmanship lies imagination. Indeed, mastery and imagination go together. This is especially the case in programming, as software is "invisible and unvisualizable" (Brooks, 1975: 185). The possibilities of what *can* be done with programming languages seems not to be the main problem; rather, what limits programmers is their imaginations. For example, in the interview with Andri in Malmö, I asked him what the biggest problem was with computers today in terms of what he thought they should do that they didn't do. His answer was rather telling:

> So far, they have been doing everything I want them to do [laughing]. I would say that perhaps *the potential of computers is bigger than people's ability to use them*. [emphasis added]

I asked if there was more processing power than we can grasp or whether we don't ask the right questions, lack imagination, or if there was some other reason:

> We can grasp it, but we cannot apply it for our own good… *there is so much more that we could do*. I think there are several aspects. First of all, there are many people who don't know what they can do with computers, so… applying computers in various areas that aren't connected to information technology right now, such as better food services, industrialising music, and stuff like that. That is probably ignorance, that *people don't know what they could do with computers*, or perhaps a lack of resources, time, and imagination. [emphasis added]

There seems to be a critique of human imagination inbuilt in Andri's argument, that there is only so much algorithms and computers can do. So, I asked Andri whether we just need to be more imaginative:

> There are definitely things we should invent and could invent. *The level of tech is sufficient to apply to much more than we do right now.* [emphasis added]

This theme was further elaborated in the interview with Bart when I asked him if there were any limits to what he could do with a programming language. He didn't think so. He thought we were just on the cusp of data analytics, of the Internet of Things, and actually using software in the world: "We have no clue of what we want to do with it. […] We can do so much more with it. […] There is really no limitation to it". He then underlined that the problem was that people are "not imaginative enough yet to actually figure out what we can do with it". Programmers, it seems, value visionaries.

The ability to envision things has always been a wizardly characteristic, and a programmer is someone who has visions of the future and how things should work, and who tries to make it happen.This imagination precedes the mastery of computer programming. Before programmers give life to the impossible, they must imagine it: "I saw the potential of what a machine could do, I saw all these possibilities", as Gabriela in Sao Paulo explained. At Facebook headquarters, I talked to Martin about his heroes, and one of them was Facebook CEO Mark Zuckerberg. I asked him why, and he answered, "because, he is a visionary". Nadja talked about one of her role models in a similar manner, as someone who was "just an amazing visionary". Similarly, according to Bart, the investor and tech entrepreneur Elon Musk "has a remarkable imagination and he is great visionary".

Wizards of the web

What is the *function* of magic? It signals the possibility of the impossible. Just like Prometheus helped humans with his Titan powers, programmers want to solve problems with their mastery of programming languages. The magic of programming today is not primarily about getting lost in a virtual dimension and forgetting about the shortcomings of the physical world; it is about using skills in programming languages to make this world a better place. Therefore, I suggest that "wizard" is an accurate label for understanding programmers today.

But what is a wizard, really? A wizard may refer to either an individual who is very skilled with computers, or a feature that guides the user through the installation or setup of a software program or hardware device (Computer Hope, 2017). A wizard is therefore sometimes also labelled a set-up assistant. Similarly, Thomas (2002) defines wizards as programs that automate complex tasks. Indeed, tasks that are complex, infrequently performed, or unfamiliar may be easier to perform with of the help a wizard.[6] According to Brooks (1975), one of the reasons why programming is fun is the pleasure of making things that are *useful to other people*: "deep within we want others to use our work and find it helpful" (Brooks, 1975: 7). He also writes that the purpose of a programming software is to make a computer easy to use.

Helping others and creating a better future through their skills in programming languages are defining characteristics of web wizards. By having this helping and assisting attitude, wizards are also expected to learn from how users use programs and anticipate what they may want to do next, guiding them through more complex tasks by structuring and sequencing them. In this sense, "wizard" also alludes to the importance of user experience (UX) in computer programming. For example, when I interviewed Mark in San Francisco, he talked about the importance of usability in his previous job as a programmer for a transport sharing company. It was about the service feeling "slick" and having such "smooth experiences" that users shouldn't notice the system that was behind the service. To be of service and help customers was especially important for freelance programmers. Python in Berlin explained how his job was interesting and challenging because he "can be part of the improvement process". In the interview with Sam in Silicon Valley, he talked about how he admires people who make things happen that change people's lives. This helpful aspect is also underlined online when googling the term "software wizard":

6. The use of the term wizard for set-up assistants started with the launch of Microsoft Publisher in 1991 (Wikipedia, 2021f). As it was targeted at non-professionals, Publisher's "Page Wizards" provided a set of forms to produce a complete document layout, based on a professionally designed template, which could then be manipulated with standard tools. Before the 1990s, wizards were expert computer users who could install software or help users with their installation. To help others is of pivotal importance, as also underlined in the etymology of the word, from the English word "wise".

> A software *"wizard"* is like a *genie of the lamp*, or a *fairy godmother*. You just say what you want, and – Hey Presto! It's done! You're spared all the complexities of wrestling with loads of complicated configuration options that you may *never* be interested in. You certainly don't care about such things when you first install the system, since you don't yet know enough to even have an *opinion* on many of the settings [emphasis original]. (Stack Exchange, 2021)

Wizards are prolific in tech culture. General Magic's early version of the smartphone (the one that came 14 years before the iPhone) was, for example, called the Sharp Wizard. People at General Magic were perceived of as wizards, with reference to the famous *Wizard of Oz* (see Fisher, 2018). Similarly, Thomas (2002: xv) writes in his foreword that "hackers perform high tech acts of wizardry", and Nathan Ensmenger (2015) has equated wizards with gurus, computer boys, or even high priests of new technology. Rosenberg (2008: 23) writes about software as "crafted by individual wizards or small bands of mages", and Frank Pasquale (2015: 86) asks rhetorically whether programmers really are "the Gandalfs of the digital world, wizards, selflessly guiding us through digital brambles". And finally, the preface to Levy's (1984: xi) book includes a list of individuals in the book with the title, "Who's Who: The Wizards and Their Machines".[7] These are people who allegedly loved the magic of the computer and worked to liberate it so it could benefit all people.[8]

The self-identification as wizards is also interesting to note; I first came across this at one of the social meetup groups I joined called "Wizard Amigos". They considered themselves a free open source, open collaborative, and free DIY (do it yourself) programming school. According to themselves, they "are a community of self-employed nomadic developers who collaborate on projects, share skills and build awesome products and services" (Wizard Amigos, n.d.). Many of the meetups I attended chose to label their members as wizards, such as the Berlin School of AI meetup, consisting of 598 "wizards". Wizards show up in other domains as well. A quick Google search and I find various design companies such as the UK-based Wizard of the Web, an Indian web development company named The Wizard Group, a blogger who calls himself *the* wizard of the web, and "a team of digital alchemists", a professional collective of web developers, designers, photographers, and "code wizards".

Wizards as helpers resonated in almost all of my interviews. "We are a tech team that helps others", as Anders, a programmer at the Daily, described himself

7. Of the 65 characters listed, only 2 are women, 9 are machines, and the rest are men.
8. One important person dubbed a wizard was the American inventor Thomas Edison. In 1878, he built a laboratory in Menlo Park where, among other things, he invented the phonograph and electric light. This was like magic to people. Hence, Randall Stross's 2008 biography of Edison is titled *The Wizard of Menlo Park*. As people today invest hopes in technology and programmers to solve all kinds of problems humans are facing, Edison's name also became emblematic of the wonder and promise of the emerging age of technological marvels.

and his team's work. Python stated that he "doesn't want to sell people stuff they don't really need. I want to help, solve problems". This was further underlined in the interview with Bart, when I asked him about his heroes and role models; one of them was Microsoft founder, developer, investor – and perhaps most importantly here, philanthropist – Bill Gates. Why? Because he was "changing the world for the better", trying to cure HIV, AIDS, and malaria. Bart went on and claimed that these folks were the ones "that actually are going to help us with climate change – not the government". This resonates with the techno-missionaries discussed previously (see Chapters 4 & 7), and how Martin at Facebook proudly proclaimed that his technology helped in the fight against ALS and made the world a better place: "We helped others, and that was very rewarding". In this way, wizardry also becomes connected to problem-solving and solution creativity.

Some researchers consider it a technological fetish to believe in digital technology to magically solve all problems (see Comor & Compton, 2015). Indeed, magic provides a vocabulary to talk about things that cannot be fully understood. According to Chris Dancy, "people resort to the occult in response to the new surroundings of electronic information". It is when technology is fetishised that programmers are transformed into magicians, people who make the impossible possible. For example, Ted, the app programmer in Malmö, complained about customers coming in and asking for an app and the price without any idea what it should do or on the basis of what data. Another example is the programmers and journalists at the Daily; while the in-house programmers at the newspaper were interested in journalism and had some kind of basic understanding of the profession, it was not always the other way around. The programmers I interviewed at the Daily felt that journalists underestimated how difficult it would be to program certain functions. Journalists treated their in-house programmers as magicians who could just cast a spell of code, and the functions journalists wished for would magically appear. This was exemplified in an interview with Daniel:

> If someone who is *not* technically knowledgeable should think about a solution – "There should be a button, and when you press the button, exactly the stuff you like should come up" [mimicking a journalist] – and then you ask, where does the stuff you like come from? And he answers, "from pressing the button". But if there should be a button that selects articles you like, then there must be a model behind it, some really advanced AI. [emphasis original]

Hence, the wizard identity is both dependent on self-identification, but also on outside projections. We, who are stuck in the physical world, often project wizardly capabilities onto programmers when fetishising technology. Through their skills in computing, programmers are able to help us solve a myriad of problems in more or less magical ways. In this manner, it is possible to connect

tech solutionism to magic. Nagy and his co-authors (2020), for example, refer to tech as a magical panacea for all problems in almost all walks of life. Tech is the trick that makes a problem disappear or become solved.

While sometimes annoying for programmers, this fetishisation of technology has its benefits and also feeds into programmers' own identification with the magical. That many users perceive technology as alien, unfriendly, and hard to understand supports programmers' self-image as geniuses doing extremely complicated stuff. "I say I am like a wizard; I make the things appear on the computer that you want to appear", as Hans in Malmö answered my question about how he would describe his profession, surrounding his programming practices with an air of secrecy and mystery.

But a wizard doesn't keep mastery of programming languages a secret; a wizard helps others with their problems, as stated earlier. In Berlin, Python underlined the importance of solving others' problems when explaining why he quit his consultant job and became a freelancer. I asked Python what he liked most about being a freelancer, and if it was specifically about solving problems:

> I'm curious, and for me, it is an amazing thing to go into a company that I don't know anything about, and then people start telling me how they work, what the software has to do, and how the processes are. It's like analysing the company and finding the perfect solution for that company. […] Solving their problems, yes. It's a lot about asking, it's a lot about understanding how they function: how does their work function and what do they need to make their life easier?

It is possible to connect the idea of a problem-solving wizard to fortune telling, with oracles and shamans looking into the future using a crystal ball or intoxicating fumes, as with Pythia, the famous Oracle in Delphi. It is by no coincidence that one of the algorithms at the Daily – automating the decision of which online news stories should be locked (in order to convert readers to paying subscribers) – went under the nickname the Oracle. Indeed, to help others, to create usable software from a UX perspective, often implies an ability to predict users' intentions and to anticipate their behaviour. When I asked Sam in Silicon Valley how he would describe his job for his mother, he answered that he works with "helping users get better and more relevant search results by better predicting users' intentions or helping users understand their own intentions". In data societies, instead of crystal balls or intoxicating fumes, there are predictive logics of automated systems, using data-rendered traces from the past to predict future behaviour (as also touched upon in the previous chapter). In other words, a wizard isn't only someone who helps, but also someone who (at least in theory) has the abilities to predict the future and anticipate users' needs and intentions. Then, a programmer become a "whiz", as some pioneers in Silicon Valley labelled smart people (see Fisher, 2018).

From hackers to wizards

The dark, secret, and slightly criminal connotations of the hacker have evolved into a more benevolent wizard, as "someone who helps others in some process or in some work", as Andri, the Estonian programmer in Malmö, expressed it. As Christopher Steiner (2012) points out, the hacker has always had an ambivalent meaning as someone who is both inherently criminal and as someone skilled at computer programming. The hacker is an outsider, a mysterious personification of technology. Or in the words of Thomas (2002: 54), "the technological is endowed with an almost uncanny sense of mystery, a kind of informational alchemy in which the hacker is able to convert gibberish into data and data into secrets".

But, according to Levy (1984), something was missing for hackers: the physical "real" world. The transformation to wizards is about bringing the real world in, not losing touch with the physical world, and helping others; if only people could get their hands on computers, the world would be a better place, with the magic in the machine unleashed. Hence, the push for personal computing in the 1980s (see Chapter 3) can also be connected to the transformation from hackers to wizards. Developer Bruce Horn claims that "personal computing is the most powerful way to extend individual human ability" (in Fisher, 2018: 123). With the widespread proliferation of personal computers (PCs), for some hackers, the war was won and their mission completed.

But everyday computer users were not as skilled in computing as hackers and did not have the same understanding of the machine. And as the computer developed with increasingly smooth user interfaces, they didn't *need* to understand the workings behind the screen. Sherry Turkle (1995) has written about this in terms of a culture of simulation and simulation aesthetics, allowing users to stay on the surface, not needing tech knowledge or understanding of the inner workings of the computer. According to her, this came with the introduction of the Macintosh and the personalisation and popularisation of computer use. The Macintosh interface gave users permission to stay on the surface; hence, the differentiation between the user in front of the screen and the programmer (or hacker) behind the screen became accentuated (see also Mansell, 2012).

With the PC and more popular access to computer software, a new generation of hackers who no longer found it necessary to break the law was born (according to Thomas, 2002). This is in line with Zuckerberg's (2012) understanding of hackers:

> The word "hacker" has an unfairly negative connotation from being portrayed in the media as people who break into computers. In reality, hacking just means building something quickly or testing the boundaries of what can be done.

In a way, it's possible to argue that Zuckerberg attempts to rebrand the hacker as a wizard here. What unites the hacker and the wizard is perhaps the pleasure of exploration and breaking new ground. And being wizards, it is not enough to imagine a better world; being wizards, they must also make it happen.

The trajectory from the hacker to the wizard also relates to the tension between perceiving programmers as a boyish prankish elite versus techno-missionaries driven by ethical motivations (see Chapter 4). The pranking dude is in contrast to the more benevolent wizard, someone seeking to contribute to a better future, to help the computer illiterate in front of the screens, and thus driven by some kind of ethics. For some programmers, it is still motivating to do things just because they can, for the fun of it. However, a wizard is predominantly a helper, a problem-solver, and someone who creates a better future through imagination, skills, and mastery of programming languages.

▼▼▼

In this chapter, I attended to the mythical and magical in tech culture. Programmers stand between the physical world and another dimension of data. And they use their knowledge, skills, and access to this other dimension to make magic in this world. With a focus on helping and making the world a better place, together with an inclination towards magical symbolism, "wizard" becomes the perfect label to capture the role of programmers in contemporary data societies. This doesn't mean that these wizards of the web always get everything right, or that there are not unintended consequences when they set out to help users with code and technology, as exemplified in the case of predictive policing and with the practice of data-driven development.

In the next and final chapter, the focus is on how this special kind of magic, mathemagics, can become evil, as it is connected to calculations and correlations of large datasets into patterns. With the help of sixteenth-century theologian Giordano Bruno, I further discuss the evil of tech as well as suggest what is needed to make the world magic again. Hence, the last chapter is an outlook – more speculative and less empirical.

Chapter 9

Summary & outlook

In this final chapter, I first provide a summary of the book. The remainder of the chapter is best described as an outlook, I connect the theme of magic to some of the larger macro discussions of power that I took as the starting point in the Introduction, justifying this study of programmers and tech culture. Being based in binary code, programming is basically mathematical magic, or "mathemagics". Mathematical magic was discussed already in the sixteenth century by theologian Giordano Bruno, who is introduced in the first section together with the mathemagician, a performer doing tricks on stage using numbers. Bruno's teachings can be used also as a critique of data societies and for a normative argument against the rigidity of computers. Hence, in the second section, I apply Bruno's warning of an *evil* side of mathemagics to contemporary data societies, but also suggest that it is possible to make the world magic again. I continue by arguing that tech culture is both modern and magical. And in the subsequent section, I underline the importance of having bodies that eventually die and decay; there would be no magic of life if humans were immortal. I started this book by claiming that programmers increasingly possess power in contemporary data societies; in the final section, I conclude that this power to a large extent is situated in programmers' imaginations, and it is possible, as well as desirable, to open up tech imaginaries to a larger section of the population.

Summary

The aim of this book has been to shed light on programmers, who they are, and where they come from. Who are they? They are the wizards of the web. This is the somewhat unlikely outcome of the odd mix of hippies and hackers, lately influenced by entrepreneur and startup ideals within – until now – a mostly young, male, suburban, and Western middle-class. They are benevolent, yet disruption-seeking, and optimistic in their aim to make this world a better place. They are rock star entrepreneurs at the same time as they are insecure programming nerds.

The symbols and labels in tech culture indeed revolve around the mythical and magical (such as unicorns), but also around symbols like the beanbag,

emphasising the more laid-back and entrepreneurial values gaining ground in the culture. And then there is the ubiquitous presence of Lego, symbolising the value of curiosity and the practice of tinkering. Both science and fiction are important here, skills and imagination, the artistic as well as the logical and rational, as epitomised in the Prometheus symbol. In my interviews, the ones talked of as good examples were often the interviewees themselves, the geek geniuses, the cowboy coder, the hacker or disruptor thinking outside the box, or underdogs taking on corporate giants. At the same time, Silicon Valley has transformed from the underdog to the mainstream, from the avant-garde to the centre of tech culture, and for some, the centre of the whole wide world. The hacker value of free information sometimes seems far away in today's tech culture, but it has left its mark. Among the heroes, I found the philanthropist who went to poor countries to set up networks and help solve problems through his knowledge. Indeed, the wizard is someone who helps others by mastering technology and having access to another dimension of ones and zeroes. If wizards become rich doing so, no one seems to raise an eyebrow any longer.

Rules were discerned by observing and interviewing how the work was conducted and digging deeper into the different expectations when programming and writing code. I emphasised the importance of grit, not giving up, as well as the expectation and pleasure of flow, to be able to spend long (enjoyable) hours in front of the computer. I also underlined the importance of the trial-and-error approach of data-driven development, rolling software out little by little, seeing how it works, getting feedback in terms of data, and then improving it accordingly.

Values guiding programming practices and tech culture were studied by observing and asking about what education, skills, and characteristics are deemed necessary to become a good programmer. I underlined that curiosity and learning were held in high regard, while formal education was generally frowned upon. Geek geniuses are often self-educated (or so they like to believe), and for this, grit is needed. One value attended to was solution creativity, the importance of being creative and not seldom solving problems in spectacular ways, that is, thinking outside the box. Other values discussed were progress, an optimistic belief in a better future, and the ideals of disruption and innovation – values about the solutions rather than the problems. In terms of imaginations, I highlighted a profound belief in data and the predictive possibilities of patterns believed to be hidden in data.

To conclude, we in front of the screens often ascribe magic to machines and view programmers as benevolent wizards, helping people solve seemingly unsolvable problems. Programmers bring the magic out of the machine, and for that they need imagination. They must be visionary and creative as well as masters of the art of coding and programming languages. As wizards, they help users in the so-called real world with their access to this other dimension

of ones and zeroes which they master through their knowledge and skills. Instead of magical wands, these wizards use programming languages to make the seemingly impossible possible.

Mathemagics

At SXSW2019, I found myself at a panel on AI and bias. In the panel, it was stated that people in front of the screens tend to think about AI as magic, though AI is really just statistics, "really advanced mathematics". However, must magic and mathematics be in opposition to each other? It turns out that the answer is no.

The following day, I listened to Cassie Kozyrkov, head of Decision Intelligence at Google, and this was when I heard the term mathemagics for the first time. In her talk, Kozyrkov warned against thinking of data as magic, and she used the term to highlight the danger of when we in front of the screen mystify what goes on behind it. While agreeing with her, this wasn't the main reason I became intrigued by the concept. For me, the seemingly unlikely marriage of mathematics and magic is a perfect description of programming, its role in contemporary data societies, and how code and digital technology are perceived by users. As attended to in the previous chapter, the magic of tech consists of how something seemingly lifeless is animated with the application of programming languages. It is binary machine code consisting of ones and zeroes that search and find hidden patterns in so-called Big Data, patterns which are believed to have explanatory as well as predictive power. It thus seems that the magic of tech isn't just any kind of magic, it's mathematical magic, the magic of science in combination with fiction, creativity, artistry, and thinking outside the box. I noted to myself that mathemagics is an excellent concept that goes to the heart of tech culture and programming, with all its allure and contradictions.

Commenting on the promise of AI, Kozyrkov argued that it is just a tool, and that people should use it responsibly. She further blamed science fiction for having given a distorted image of technology, and said that she dreams of a demystified "boring and useful" AI. Thus, by emphasising how magic is a front-stage projection, nothing more (or less) than a mind-blowing demonstration influenced by sci-fi imaginaries, it seemed to me she was trying to take the magic out of computer programming. I agree with her claim that tech is not autonomous and separate from the people behind it – indeed, this is the rationale for having written this book – however, I'm not willing to take the magic out of tech.

Arguably, the first one who thought about mathematics as magic was the Italian Dominican monk, philosopher, mathematician, poet, cosmological theo-

rist, and occultist Giordano Bruno (1548–1600).[1] For Bruno, there were three sorts of magic: divine, natural, and mathematic (see Bruno, 2018, his collected translated works). The three sorts of magic were associated with three different worlds: divine magic is in the archetypal world; natural magic resides in the physical world; and mathematical magic is found in – not surprisingly – the rational world.

When it comes to mathematical magic, Bruno argued that what he called a spirit magic and standard mathematics are both ways of interaction and manipulation. Mathematic magic locks the world, and things in this world, into symbolic meanings, "symbols that mean something", as Scott Gosnell phrases it in his introduction to the translated works of Bruno (Bruno, 2018: 15). According to Bruno, imagination, rather than physical senses, is needed to make sense of symbols. In other words, mathematics becomes an imagination that interacts with people and has bearing in this world that people inhabit. Bruno described this as halfway between the natural and supernatural (divine). In this way, it is possible to translate Bruno's tripartite delineation of magic to magic in tech: there is divine magic, which is about making the impossible possible; there is natural magic of physics and logical thinking; and then there is the mathematical magic, which is a marriage of the first two:

> Mathematical magic is the mean between divine and physical magic, just as simple mathematics is a midpoint between natural and metaphysical. (Bruno, 2018: 65)

Bruno came very close to ideas that would resonate with hippie pioneers and the theory of Cybernetics centuries later. Because, like hippies, Bruno meant that the divine was omnipresent and immanent and resides within all humans, as he considered humans to be created in the image of God and also to have souls. He talked about a universal spirit "which is totally in the whole, and each part of the world" (Bruno, 2018: 33). This universal body, which is present "in the whole and everywhere" (Bruno, 2018: 49), is similar to ideas of information connecting all living beings on Earth as well as connecting humans to a higher entity (even though not all hippies talked about a god). Bruno underlined that this doesn't mean all humans possess God-like qualities; still, his teachings were too difficult to accept for the Roman Inquisition, who condemned him to death for heresy.

Bruno also had something to say about the ones performing the magic. For him, the magician was someone who used the intertwined nature of the universe to influence it by using *creative* reasoning. Indeed, creativity and art were important also for Bruno, and because of their ability to influence people's

1. Bruno was tried for heresy by the Roman Inquisition because of his teachings of, amongst other things, reincarnation, pantheism, and a heliocentric solar system. He was eventually found guilty and burned at the stake.

minds and bodies through their senses, Bruno even claimed that artists, poets, and painters may be doing magic. Thus, in Bruno's view, the magician was an intermediary between God, God's cosmos, and humanity, manipulating these forces for the greater good. In other words, for Bruno, the magician is also a wizard, a benevolent person who stands between different worlds.

A magician, according to Bruno (2018: 23), is first meant to refer to a wise man who, through a better understanding of the universe, is more equipped to affect change in it: "a magician means a wise man with the power of action". It is interesting how Bruno underlines action beside wisdom. It is not enough to have knowledge, or in the case of programmers, the ability to master programming languages. The magician also *acts*, does something, plays hockey rather than engaging in endless talking (as discussed in Chapter 5).

To my knowledge, Bruno did not talk about mathemagicians. This label was probably first applied to mathematics and popular science writer Martin Gardner (see Berlekamp & Rodgers, 1999), who is regarded as an important magician due to his interest in magic and illusion. Gardner's first and perhaps most famous book, *Mathematics, Magic and Mystery* (1956), was about mathematically based magic tricks.[2] A Google search for the term "mathemagician" results in a plethora of images of magicians and illusionists on stage, not seldom in cloaks decorated with visible numbers and equations. The term suggests that mathematics can be used in magic performances, often involving mentalist tricks – such as clairvoyance and abilities to forecast the future, sometimes with claims of being connected to divine or supernatural agency (so-called divination) – all relying on mathematical principles. In other words, a mathemagician is a mathematician who is *also* a magician (Wikipedia 2021c). In the same way, a programmer is someone who is versed in numbers (here programming languages), and with this knowledge, performs magic and makes the impossible possible.

The Mathemagician in *The Phantom Tollbooth*

Perhaps the most famous mathemagician is the character in the 1961 children's fantasy adventure novel, *The Phantom Tollbooth*, by Norton Juster, which tells the story of a bored young boy, Milo, who receives a magic tollbooth that transports him to the Kingdom of Wisdom. The ruler of Digitopolis, the Kingdom of Mathematics, is the Mathemagician. The kingdom was once prosperous, but is now troubled because of a fight between the Mathemagician and his brother, King Azaz the Unabridged, ruler of Dictionopolis. The land is thus divided between two brothers fighting about whether numbers or words

2. Here, the mathemagician is a performer, entertaining live audiences with mental math capabilities. While fascinated by how physical laws were seemingly violated in illusionist tricks, Gardner was nonetheless a fervent sceptic and missioned against what he called pseudo-science (not least in his book from 1957 with the title, *Fads and Fallacies in the Name of Science*). He promoted scientific inquiry and the use of reason in examining extraordinary claims.

are of most importance. Previously, all lived in harmony because the brothers' adopted younger sisters, princesses Rhyme and Reason, settled disputes in the kingdom. But when the princesses decided that letters and numbers were of equal importance, they managed to outrage both the Mathemagician and King Azaz, who banished them to the Castle in the Air. Since then, the land had neither rhyme nor reason, and Milo therefore goes on a quest to restore the kingdom by returning its two exiled princesses.

This is a wonderful adventure story full of puns and wordplay (such as when Milo unintentionally jumps to Conclusions, an island in the Kingdom of Wisdom), permeated by a love of education as well as highlighting the perils of manipulation through words and numbers if rhyme and reason are lost. Indeed, the book has much relevance in today's crazy world of fake news and election tampering via social media platforms, as also underlined by Maurice Sendak in an appreciation in the 35th anniversary edition of the book (Juster, 1961/1996). The book follows in the same tradition of Martin Gardner (contemporary to Juster) who was also interested in wordplay, logic, and reason. Indeed, towards the end of the 1950s and beginning of the 1960s, there was a burgeoning interest in the magic of mathematics. Just two years before *The Phantom Tollbooth*, a 27-minute Donald Duck educational film was released, *Donald in Mathmagic Land* (Clark et al., 1959), which was nominated for an Academy Award. Donald is initially not interested in exploring Mathmagic Land, believing that math is just for so-called eggheads. But when it is suggested to him that there is a connection between math and music, Donald becomes intrigued. For Donald Duck as well as Milo, it is through education that they develop and learn to love things that previously bored them.

Bruno, making the world magic again

Bruno warned against a possible evil side of mathematical magic, his normative critique remaining remarkably relevant in contemporary data societies. His call for not abandoning the magic metaphor is also interesting, how the world of tech can be magic without becoming evil. He specifically underlined how important it is "to account for all the miraculous things that are in nature" (Bruno, 2018: 14), and perhaps also the magic within people themselves. As we lose ourselves in rules and regulations, we also lose contact with the specificity of our circumstances, the magic of the unique situations and circumstances we find ourselves in. Poet and writer Helena Granström, who appeared on Swedish radio (Sveriges Radio, 2020b), reflected on the meaning of the Christian holiday Ascension and emphasised that even if many in a secular world do not believe in the miracles of Christianity, it is still possible to believe in the miracles that occur *within*. In a similar way, Bruno attempted to make the world magic again, through presenting the nature around humans, as well as ourselves, as living and unpredictable.

But for this magic to happen, it is important to avoid trying to capture or fix it into rigid categories and regularities and to accept that there will always be unknown things, which is what prompted Bruno to warn against the risk of mathematical magic becoming evil.[3] It is easy to draw parallels to Cathy O'Neill (2016) and her account of *Weapons of Math Destruction* and the perils of predictive policing, as attended to in the Introduction of this book. A smokescreen of mathematical symbols in societies with a pervasive desire for numbers (see Kennedy, 2016) gives an allure of objectivity, neutrality, and fairness; this is one reason why it tends to be unquestioned when mathemagics is directed at us. According to O'Neill (2016), such smokescreens are why predictive policing is believed to be scientific, fair, and largely accepted, when it is in fact biased. Numbers and data have an allure of objectivity and neutrality, a higher and more scientific knowledge, and thus may appear as magic.

Philosopher Jonna Bornemark, to whom I also listened to on Swedish radio (Sveriges Radio, 2019), argues that instead of letting the computer enslave us, we should use it for our purposes; we should approach the computer from a humanist worldview, not approach humanity from a computational worldview. Bornemark underlines how the *human factor* is often blamed when things go wrong; examples of when this regularly happens are when airplane crashes or boat accidents are to be explained.[4] However, Bornemark (2018) argues that the problem is sometimes the *rigid factor*: when the computer is not working intelligently enough, it is sometimes because it is too restricted by automated systems and rules of algorithmically predefined decision-trees (as Dreyfus argued already in 1972). It is easy to relate to *Little Britain*'s sketch, "Computer says no", in which character Carol Beer always responds to a customer's enquiry by typing it into her computer and responding "computer says no" to even the most reasonable of requests (Walliams & Lucas, 2003–2007). As Frederick Brooks (1975: 164) explains, a computer program is a message from a programmer to a machine, "to make your intention clear to the dumb engine". Vikram Chandra (2013: 5–6) similarly argues that "code is […] a series of commands issued to a dumb hunk of metal and silicon and plastic animated by electricity". But it is not one-way communication (where commands from a human are to be understood by a machine); the program (machine) in turn must communicate in a way that makes sense to human users. If programs would function more like humans, there would not be so much work debugging them, and computers would not say "no" that often.

To rely on information stored on – or generated by – a computer, and then make decisions based on that according to fixed calculation rules, even in the face of common sense, is an example of a computer's rigid factor. Because common

3. This also comes very close to Weizenbaum's (1976) critique of mechanical magic.
4. A prime example is when Italian cruise ship Costa Concordia ran aground in 2012, resulting in 32 dead, and how the investigation focused on the shortcomings of the very human captain (who was sent to jail for manslaughter).

sense cannot be coded into a computer – not yet, as it seems. Ethiopian Airlines flight 302 (attended to in Chapter 1) most probably did not crash because of a human factor, but because of a *software factor*. Tellonym's algorithms and filtering system (also addressed in Chapter 1) need human input, because the rigidity in their agency fails to deal with the messiness of the social context in which communication between teenagers takes place, confusing bullying and hatred with honesty. An infamous example of the need for humans is when Facebook fired its trending team, which resulted in the algorithm trending fake news and things such as a man masturbating with a hamburger.[5]

Programming languages are also problematic in that they are fixed and rigid. In programming languages, following mathematical Boolean logic, everything can be reduced to binary code, truth and falsehoods, ones and zeroes. But this strips humans and the physical world of complexity and irrationality. Thus, following Bruno, part of the magic of life is the unpredictability that so-called beautiful code should avoid (see Chandra, 2013). Is it really possible to capture the beauty of art and poetry in the inscribed, fixed, logical, and universal rules of programming? Too much inscription, fixity, calculation, and measuring can become evil. As Evgeny Morozov (2013) argues, a blind faith in rationality (or efficiency, as he chooses to label it) reduces the human, which is in agreement with Joseph Weizenbaum's (1976) argument 45 years ago: according to Weizenbaum, an engineering style of thinking trivialises human life.[6]

Those who are too focused on rules and regularities, and are completely governed by mechanical thinking and logical rigidity, are labelled pedants by Bruno (in Bornemark, 2018: 47). One example of such pedantry is when Bob Saunders (a famous hacker presented in Levy's 1984 book) was asked by his spouse Marge if he "would like to help" her bring in the groceries, to which Bob answered "no". When Marge got angry, Bob answered "I don't like to, but if you ask me to help, I do it" (Levy, 1984: 25–26). Indeed, in order for spells to work correctly, they must be uttered following a precise order, otherwise the machine won't understand, leading to "computers saying no".

The mathemagician as a pedant

As with Bruno's pedants, the Mathemagician in *The Phantom Toolbooth* (Juster, 1961) is also completely rigid. To give an example, Milo asks the Mathemagician to show him the biggest number in the numbers mine; the Mathemagician brings out an enormous 3. This is obviously not what Milo meant, who then

5. The so-called mechanical Turks were born out of this failure of rigid automated systems, referring to 48,940 people in 167 countries sorting and labelling images (see Bunz & Meikle, 2018).
6. Given the experiences he had with people interacting with his chatbot Eliza, Weizenbaum was sceptical about the integration of computers into society and also foresaw the dark side of human-computer interaction.

starts over and asks for the longest number. He is presented with a very wide 8, and so on. Here, the Mathemagician becomes like a rigid computer, a pedant who can't understand the meaning and context in which Milo's questions are posed. He takes the words in Milo's question quite literally and doesn't understand what Milo is seeking. For this, the Mathemagician would need to also consider the *context* of Milo's question. It is indeed easy to be wrong, as the Mathemagician informs Milo, in a way that also illustrates the frequency of bugs when dealing with rigid machines that process every query quite literally, understanding neither the context nor nuance in which questions are posed.

Already when Milo is on his way to Digitopolis, he learns that mathematics is about being precise and exact and that there are no illusions in this kingdom. But this is precisely what becomes the illusion, that everything in life is precise and exact. This is illustrated when Milo meets with 0.58 of a child, who, together with parents and two siblings, make up an average family. His reaction to 0.58 is to protest that averages aren't real, they are just imaginary. The partial child then enlightens Milo that "one of the nicest things about mathematics, or anything else you might care to learn, is that many of the things which can never be, often are" (Juster, 1961/1996: 197). This is a nice illustration of Bruno's conception of mathematics as symbols that mean something, as an imagination that interacts with people and has bearing in this world.

It is possible to draw parallels to the Quantified Self (QS) movement (see also Chapter 2), and Bornemark (2018) claims that for QS practitioners, the body loses its meaning as a living entity and becomes an object that may be controlled from the outside. However, it could also be argued that for QS practitioners, tech helps them see what they cannot otherwise see or feel, "a prosthetic of feeling" (Neff & Nafus, 2016: 75), something to help them sense their bodies and the world around them, to bring out its magic. And this is also the idea of Chris Dancy's (2018) plea to *not unplug*, that people can use code and data to get to know themselves and become more mindful. According to Chris, "tech is a magnified glass on life, not to extend or shorten it. With tech, we get to explore inside our body and inside our mind. That is the greatest gift technology could give to us". Chris argued that tech was missing an opportunity to introduce users to who they *really are*, and not only who they *want to be*.[7] Does tech contribute to escapism? Are people trying to run away from themselves? Chris seemed to think so, claiming that people are afraid of looking into the mirror that he argues technology could be, being afraid of getting too large a dose of their own behaviour. This reminds me of *The Wizard of Oz* (Baum, 1900), in which the main character Dorothy meets the Scarecrow, who wants a brain, the Tin Woodman, who desires a heart, and the Cowardly Lion, who needs

7. I have written about this elsewhere, that technology use is rather indicative of an ideal self and how we wish to be perceived (see Svensson, 2011, 2015).

courage. Dorothy invites them to accompany her to the wizard so they can ask him for help. After being exposed as a fraud, the wizard gives the Scarecrow a diploma, the Lion a medal of courage, and the Tin Man a ticking heart-shaped watch. In this way, the wizard helps them see that the attributes they sought were within them all along. For Chris, this would be the real magic of technology: to help people find themselves and the answers they already carry within them. Hence, there is no need to take the magic out of technology, as Kozyrkov attempted to do in her SXS2019 talk. Because, as Bruno argues, mathematical magic does not have to be evil, it can be good as well.

Modernity & mathemagics

Tech culture is deeply modern, with its focus on mastery and control. In the previous chapter, I underlined manipulation as a fundamental aspect of magic and used this to argue that programming is about magic rather than religion. With this focus on manipulation through mastery of programming languages, Modernity and mathemagics go hand in hand. While mastery and control of nature and the body (the flesh) are themes in Modernity, modern wizards of the web seek to control the future through mastery of programming languages. It was the quest to control the power manifested in the computer that drew programmers to the machine in the first place (according to Weizenbaum, 1976). Norbert Wiener (1948) argued that the thought of every age is reflected in its technique and that computers are all about communication and control. 70 years later, researcher Xanthe Whittaker (2018) continues to argue that contemporary digital technologies are instruments of management and control.

This form of control can be linked to the argument of digital media as existential and environmental (as discussed in Chapter 2), with Wendy Chun (2011) arguing that the opacity of digital media is connected to the disappearance of the machine. With reference to Foucault, this is connected to a form of surveilling power and control, shaping the *context* in which people operate, rather than exercising control on an individual level. In her book *Control and Freedom*, Chun (2006: 9) argues that digital languages make control systems invisible: "we no longer experience the visible yet unverifiable gaze but a network of nonvisualizable digital control".

Then there is the illusion of the control digital media gives rise to. Peter Nagy and his co-authors (2020) write about time-hacking technologies as an opportunity to orchestrate life trajectories, to use tech to control mental and biological pasts, presents, and futures. Douglas Thomas (2002) also writes about control as an illusion. While control of information was part of early hacking, in the 1990s, the world was so overwhelmed with information that control became a contested issue. Still, a belief spread that those who manage

tech would control society, in a sense equating knowledge and mastery of code and programming languages with control over society.

In this sense, magic, which for Bruno was part of nature and the natural world, is reduced in Modernity, where magic implies a violation of the idea that the world can be accurately described in terms of mathematical laws and thus controlled (see Bornemark, 2018). However, as Bruno underlined, it is possible to make the world magic again. First, it seems Modernity must be left behind, or perhaps people have never really been modern, as Bruno Latour argued in 1993. The distinction between nature and society is no longer possible (the Covid-19 pandemic and its consequences for societies is a prime example). Or, perhaps this distinction never existed in the first place.

I asked Chris Dancy about control when I met him in Austin. While reading his book, I was intrigued by the concept of a pendulum of being in control through access to data and not being in control at all. In a sense, the whole book is about how he is trying to gain control of his life and his health through data. He advocates becoming more conscious and mindful and that this can be achieved through "not unplugging" (that is, being *more* connected with digital devices, in contrast to the recent digital detox or digital disconnection movement; see Syvertsen & Enli, 2019). Still, he ends the book by stating that people *believe* they can control everything through data, while they really cannot control anything. When I asked about this, Chris admitted that he basically thinks *everything* is out of control.

The values of progress and technological solutionism are also themes which resonate in Modernity. Daniel Rosenberg (2013) associates the rise of the concept of data to Modernity, and Mark Jarzombek (2016) argues that data processing is about making the self and others predictable, identifiable, and exploitable, as in the directive of Modernity. To participate in the project of Modernity has always meant that one becomes a calculable subject (see Raley, 2013). Gideon Kunda (2006: 225) talks about tech as having a mission, to forward progress and individualism, where people work "in the name of humanism, Enlightenment and progress"; and this was why any analysis of the role, use, and social consequences of the technology developed in the company he studied was absent or glossed over with the words of "innovation, productivity and profit with their connotations of inevitability and rightness" (Kunda, 2006: 225). Indeed, the modern belief in endless progress lurks behind these optimistic ideas of a better future through the use of technology. To question tech solutionism is then like questioning the Enlightenment itself. If you are against the Internet, you are considered anti-modern, according to Morozov (2013).

Modernity and measurement also go together. The modern quest for uniqueness and exceptionalism can be understood as being behind contemporary practices of self-tracking (see Morozov, 2013). Phoebe Moore and Andrew Robinson (2015) write of a modern gaze, in terms of an observing subject (mind) and an observed object (body). The QS movement, they argue, is engaged in a

dialectic of self-observation and self-exploitation. Tech in general has indeed embraced the idea of disembodiment. I have stumbled across imaginations all over tech culture that it would be possible to upgrade, or get rid of, the physical body. In general, hackers have always had a problematic relation to the body; in popular depictions, they are overweight and unattractive, spending long hours in front of a computer neglecting their bodies (the so-called computer-bum phenomenon; see Ensmenger, 2015), but behind the computer, they are omnipotent (see Thomas, 2002). As Levy (1984) puts it, programming is the ultimate disembodied activity.

It is possible to trace this imagination to tech culture's origin in the 1960s countercultural movement and hippie influence (as attended to in Chapter 3). The out-of-body experience, induced foremost by LSD and other psychedelic drugs, greatly impacted how some pioneers imagined the future and the role of computers in it. It was believed that LSD allowed them to escape their bodies and experience a kind of shared consciousness, the same with computer mediated communication. The idea of a virtual reality thus became linked to San Francisco Bay area stories of LSD and countercultural transformation. LSD could help programmers enter cyberspace, and even if this was a dangerous place, it could be beautiful, enticing yet strange. Hence, according to Turner (2006), cyberspace became connected to acid-driven mysticism. To enter it, programmers must forsake their own bodies and *become* information. Cyberspace is generally considered a space without human bodies, with tech opening doors to another magical world in which decaying bodies will be irrelevant.

This theme also resonates in science fiction and William Gibson's novel *Neuromancer* (1984), that so-called console cowboys could wire themselves and leave their bodies behind. As I'm writing this in 2020, it's fascinating to consider that *Neuromancer* was set in this year. Disembodiment permeates the book, as protagonist Henry Dorsett Case jacks himself "into a custom cyberspace deck that projected his disembodied consciousness into the consensual hallucination that was the matrix", or how Case "lived for the bodiless exultation of cyberspace", had a "relaxed contempt for the flesh", or how he "fell into the prison of his own flesh" – for him, the worst kind of punishment (Gibson, 1984: 6; see also Chun, 2006). Indeed, the console cowboy is a computer–man hybrid and cyberspace is "a consensual hallucination experienced daily by billions of legitimate operators, in every nation, by children being taught mathematical concepts […] a graphic representation of data abstracted from the banks of every computer in the human system" (Gibson, 1984: 59).

Tech culture's dissatisfaction with the human body is a theme that resonates in Modernity. The splitting of the body from the mind has its roots in religious thinking; it was the search for a human essence, detached from the physical body, which led to the notion of an immortal soul. The body is flesh, associated with sin, and will eventually die. The soul, to the contrary, is spiritual, con-

nected to higher lifeforms, and will live forever. In Modernity, René Descartes (1647/2017) gave scientific sanction to this body–soul dualism by reframing it as a body–mind split. What Descartes argued is that the *mind* deals with consciousness and self-awareness and is the seat of intelligence, not the physical *brain*. The soul, or mind, is thus what makes us human. Indeed, Cartesian dualism implies that the mind takes precedence over the body. It is therefore possible to trace ideas of creating superhuman intelligence (see Bostrom, 2014) to older assumptions inherited from Modernity, that humans may be reduced to their minds. Dataism and data religion[8] try in a similar way to get out of the body in the quest to replace the brain with intelligent software. To make the body obsolete, and thus reach immortality, would be the ultimate trick of Modernity.

In this vein of thought, to engineer disease-immune humans might seem tempting, given our experiences with the Covid-19 pandemic. My hope was to end this book elegantly with an illustrative scene from SXSW2020, a year after the opening scene in the Introduction. However, the pandemic is pulling across the globe and has put a stop not only to SXSW and other conferences, but sports and music events of all kinds as well. While this disease is truly horrendous, it is interesting to view the pandemic from the perspective of the body–mind split. What this pandemic has shown us is that humans and the entire fabric of human societies are limited by biology; humans are part of nature, and nature is part of humans. Humans carry nature inside their bodies, as Granström phrased it (Sveriges Radio, 2020a). If only it would be possible to get rid of our organic bodies or upgrade them, then people would be protected against at least biological viruses.

However, as Peters (2015) argues, the body is the most basic of all media, and the richest in meaning; the meaning of a face, voice, or gesture cannot be captured in a thousand sentences. Nonverbal communication is clearly older than language, and emotional ties between people arguably emerge through bodies. Peters thus underlines the body as the ultimate infrastructure; it would create problems if people got rid of it. People need others' bodies, and not only in a sexual way, but also because "live presence will never lose its pull" (Peters, 2015: 276). Sherry Turkle's (2011) argument of how online connection may lead to new kinds of solitudes rings truer than ever. On the news, I heard about a report of online education during the pandemic. The conclusion was that even if it is technically possible to teach online, something disappears: the possibility of being physically present with the students. This is something I recognise myself, having had difficulties motivating students via Zoom and other online teaching platforms.

8. The belief that the entire universe consists of data flows, that organisms are algorithms, and that humanity's cosmic vocation is to create an all-encompassing data-processing system we will eventually merge with.

The magic of dying

Having underlined the importance of having a body and also being sceptical of accounts rendering bodies obsolete, this would also entail an acceptance of bodily decay and the finality of death. According to the Harry Potter documentary (Harding & Ho, 2017), magic is about trying to control something that is uncontrollable: life itself. Being a control freak myself, this is something I know all too well. But maybe this is what is really magical about life after all, that we cannot control it, that we cannot be predicted by our data exhaust. To have a body is indeed magical. The spell "hocus pocus" allegedly originates from the Latin *Hoc est corpus meum*, meaning "this is my body". But rather than transforming the communion bread to a representation of the body of Christ, my reading of the spell is to underline the magic of death; *hoc est corpus meum quod aliquando moriturus*, "I hereby present you with my body and it will eventually die". If people would ever become immortal, the magic of life would disappear.

Someone who has stressed this argument with emphasis is philosopher Martin Hägglund (2019) in his best-selling book, *This Life – Why Mortality Makes Us Free*, in which he argues that eternal life is not only unattainable but also undesirable, as it would eliminate care and passion for this life. Hägglund (2019) labels a devotion and care for a life that will end as secular faith, as he is also deeply critical of religious promises of an afterlife, which would make this life somewhat less meaningful since actions here and now would lose some of their importance. According to Hägglund (2019: 43), "the loss of your own life and the loss of what you love, is not a prospect that can be eliminated, but an intrinsic part of why it matters what you do with your life". The knowledge that life ends is what drives people to pay careful attention to what they do with it. Hägglund underlines the here and now, that actions in the present matter, which stands in contradiction to tech culture's future orientation. The magic of having a body that will die is thus connected to the magic of the uniqueness in specific situations (as Bruno, 2018, underlined in his warning of mathematical magic becoming evil).

In programmers' imaginations, it is through data that this immortality is supposed to be achieved (as attended to in Chapter 7). What I fear is that humans would only be kept alive to produce data, which increasingly powerful algorithms would mine for patterns that social media giants could use in their business models. The horror scenario isn't that I eventually will die, or that a super-intelligent AI will kill me; the real nightmare is to become reduced to a data farm with the sole purpose of producing data. A dystopic update of Descartes' proposition "I think, therefore I am" would become "I have data extracted from me, therefore I am".

Sadly, this is already happening. The PBS *Frontline* documentary about Amazon reveals that their warehouse workers are forced to wear scanners and

trackers at all times. As one of the workers explains in the documentary, "we are not treated as human beings. We are not even treated as robots; we are treated as data streams" (Jacoby et al., 2020). Moore and Robinson (2015) also discuss this monitoring, tracking, and datafication of Amazon – and also Tesco – warehouse workers through wearable trackers. They argue it is about controlling workers, rendering their work more predictable and exploitable, and recasting them as efficient, rational, masculinised, and managed subjects. According to Moore and Robinson, these quantified self workers (QSW) are pushed to quantify and regulate their own bodies to conform to an external quantified gaze. This also applies to prisoners today, who are tracked and datafied to the last detail, such as food intake, emotional status, and media consumption, in order to produce data to train machine-learning algorithms that are used in applications such as Alexa and Siri. In a recent article, media scholars Anne Kaun and Fredrik Stiernstedt (2020) conclude that rather than engaging in manual labour, prisoners today are increasingly providing and generating data to train and feed machine-learning technologies.

Indeed, today humans produce value through the data traces they leave behind in a digital existence. I have to search hard to find activities that are *not* being tracked or recorded. Mark Andrejevic (2020), for example, writes about future cities as data-collecting machines, and Whittaker (2018) talks about how technologies of measurement are linked to performance and comparisons that may reorganise work practices. Journalism is one example in which metrics have been increasingly accepted and legitimised as useful and objective measures of journalists' work. In the study of the Daily, if a news story did well online (i.e., was clicked on a lot), this was used as an indication of whether a follow-up story should be pursued. To be *thrown* into data societies (see Chapter 2; Lagerkvist, 2017) is about being subject to an external quantified gaze, a gaze which is black-boxed (in the double meaning of the world, as both tracking people and being hidden from them). We humans then risk becoming just like the prisoners in Kaun and Stiernstedt's (2020) study, producing data while having little access to the benefits this data production generates. This is a panopticon 2.0, producing data by being watched, tracked, and datafied, without knowing how this data is going to be used and acted upon us.

Chris Dancy told me when I first met him in Malmö that the big fight of tomorrow will be to allow us to die. Death has been discussed in computer–brain comparisons before. Wiener (1948), for example, concludes that a brain can never clear its past record as a computer can; only death can clear the brain from past records, but then it cannot be restarted. When writer Raul Vaneigem complains that "life quantified becomes a measured route-march towards death" (in Moore & Robinson, 2015: 13), I fear that life quantified will deprive us of death. As humans, we are not permanent, and that is what makes us special. To turn me into a data-producing machine would take the magic out of my life.

The power of imagination

I am going to end the book with imaginations. Imaginations are at the heart of cultures and a theme that cuts across this whole book. Imaginations are about power, shaping algorithmic calculations and automated systems that act upon humans in this world we live in. Indeed, the starting point for this book was an acknowledgement of the power of data, algorithms, and automated systems that run on code and software in contemporary data societies. And our mode of relating to the world and ourselves is being reconfigured through digital media. This increasing dependence on data and automated systems creates new lines of stratification in which programmers are becoming more and more influential. The programmer, like Bruno's magician, is someone who could use the intertwined character of the world, using creative reasoning to do magic. This is the magic of their imagination.

Kozyrkov underlined in her SXSW2019 talk about the perils of AI, that "the genie in the bottle isn't the problem, the problem is an unskilled wisher". What are the imaginations of what tech can do? What kind of problems can it solve? As I have shown in this book, with references to the magical, as long as programmers and users believe, they perceive no limits. SXSW2019 was, for example, full of exclamations of the hope to "create the impossible" and "do the impossible". In other words, the greatest power the wizards of the web exercise is their imagination. Bruno (2018) emphasised that magicians must pay particular attention to their imagination, because these imaginations are the root of the actions they impose on the world. In terms of today's programmers, they attempt to give life to things and functions they imagine.

Sometimes, programmers are directed and limited in their imagination by their outsourcers or the companies in which they are employed. Shoshanna Zuboff (2019) underlines the power of social media giants, the beneficiaries of so-called surveillance capitalism. As Frank Pasquale (2015) argues, black boxes embody a paradox in contemporary data societies, in which the data social media giants have access to is staggering in terms of breath and depth, while most of it is out of citizens' reach. Others have thoroughly analysed how black-boxed automated systems based on datasets act upon the already disadvantaged and limit their possibilities (such as the women, people of colour, and the poor; see Eubanks, 2017; Noble, 2018; O'Neill, 2016). This is a *negative* power that previously was approached in terms of an evil side of mathemagics, with obsessive measuring and pedants ruling the world, locking humans into rigid systems in which the human is considered second to the rational data-processing machine.

At the same time, so-called thinking outside the box is highly valued in the culture. Programmers are expected and encouraged to value their imagination, to follow up on their ideas of what data and computers could be able to achieve.

Arguably, people's digital existence, as well as their future, depends to a large extent on these imaginations. Robin Mansell (2012) has made a strong argument for this in her book, *Imagining the Internet*. How people imagine the Internet frames their understanding of it. By referencing Charles Taylor, Mansell defines imaginaries as deeper normative notions and images invoked by how people make sense of practices and how this impacts a communication system that is so central to people's lives. Imaginaries influence how digital technologies are used and the way they permeate and mediate people's lives. This gives programmers power also over users' imaginations and how people make sense of their lives in the contemporary data societies into which we are thrown.

It is therefore important to ensure satisfactory representation in terms of whose imaginations are listened to and taken into account. As I stated in the beginning of this book, it isn't likely that the power of data, algorithms, and automated systems will diminish any time soon. Hence, it is pivotal to bring in more perspectives and imaginations and seriously engage in the issue of representation in tech. Technology is too important to outsource to certain segments of the population; it must be democratised, especially to ensure that a variety of imaginations are taken into account. As Peters (2015: 8) writes, the task today is to democratise technology, "to wrest the lever from the computer nerds".

One danger here is to treat technology as unavoidable, that it will bring about certain consequences sooner or later, but it does not have to be this way. The beauty of *not* treating technology as deterministic is that the future is up for grabs – another world is possible. As Mansell (2012) claims, imaginaries are never settled and are always up for negotiation. Let us therefore open up for more people from different backgrounds and from different age groups to participate in imagining what technology should be used for and what problems could and should be addressed by technology.

There are also problems and practices that are *not* suitable for tech solutions. Morozov (2013), for example, argues that politics is one such area. Chun (2006) even claims that to think it is possible to solve political problems with technology is to respond paranoically to power. I agree with Pasquale (2015) when he argues that it is possible to imagine a future in which the power of algorithms and automated systems should be limited to environments where they actually *may* promote fairness and freedom. As Weizenbaum (1976) explained already 45 years ago, the question is not so much a matter of what computers *could* do, but what they *should* do. This is where tech could become magical in a *positive* way, considering different kinds of possibilities while at the same time retaining a critical mindset, not resorting to the pitfalls of technological solutionism.

The programmers I met during my journey into tech culture were friendly, sympathetic, and forthcoming, though this doesn't mean that some aspects of their culture never result in negative consequences. In a culture that underlines speed, data-driven trial-and-error development, values disruption, and never

giving up, things sometimes go wrong. To democratise imaginations of future tech is about opening up tech, underlining the importance of greater representation, making space for more kinds of imaginations, while at the same time appreciating the many positive aspects of the culture. And change should be possible given the value of progress and the importance of entrepreneurship. As attended to in Chapters 3 and 4, in the early days of computer research within the US military, collaboration was possible across disciplines because of curiosity and a value and respect of others' expertise and differences. As underdogs and outsider positions are held in high regard in the culture, tech should, at least in theory, be open to actively seeking and bringing in different kinds of people and perspectives in pursuit of a better future.

As it goes in the old Beatles song, "I am the Walrus": "I am he as you are he as you are me and we are all together". And, if we're all in this together, then tech and programming is simply too important to be left in the hands of a few. In other words, tech *for the good* of humanity is inevitably tech *by the many*.

References

Altheide, D. L., & Snow, R. P. (1979). *Media logic*. Sage. https://doi.org/10.1177/089124168201000412

Ananny, M., & Crawford, K. (2015). A liminal press. *Digital Journalism*, *3*(2), 192–208. https://doi.org/10.1080/21670811.2014.922322

Anderson, C. (2008, June 23). The end of theory: The data deluge makes the scientific method obsolete. *Wired*. https://www.wired.com/2008/06/pb-theory/

Andersson Schwarz, J. (2019). Umwelt and individuation: Digital signals and technical being. In A. Lagerkvist (Ed.), *Digital existence: Ontology, ethics and transcendence in digital culture* (pp. 61–80). Routledge. https://doi.org/10.4324/9781315107479-3

Andrejevic, M. (2020). *Automated media*. Routledge. https://*doi*.org/10.4324/9780429242595

Asp, K. (2014). News media logic in a new institutional perspective. *Journalism Studies*, *15*(3), 256–270. https://doi.org/10.1080/1461670X.2014.889456

Aydin, C., Woge, M. G., & Verbeek, P.-P. (2019). Technological environmentality: Conceptualizing technology as a mediating milieu. *Philosophy and Technology*, *32*, 321–338. https://doi.org/10.1007/s13347-018-0309-3

Barbook, R., & Cameron, A. (1996) The Californian ideology. *Science as Culture*, *6*(1), 44–72. https://doi.org/10.1080/09505439609526455

Baum, L. F. (1900). *The wonderful wizard of Oz*. George M. Hill Company.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for agile software development*. https://agilemanifesto.org

Beck, U. (1992). *Risk society: Towards a new modernity*. Sage.

Benjamin, R. (2019). *Race after technology: Abolitionist tools for the new Jim Code*. Polity Press. https://doi.org/10.1093/sf/soz162

Berlekamp, E. R., & Rodgers, T. (Eds.). (1999). *The mathemagician and pied puzzler: A collection in tribute to Martin Gardner*. A K Peters/CRC Press. https://doi.org/10.1201/9781439863848

Bogost, I. (2015, January 15). The cathedral of computation: We're not living in an algorithmic culture so much as a computational theocracy. *The Atlantic*. https://www.theatlantic.com/technology/archive/2015/01/the-cathedral-of-computation/384300/

Bornemark, J. (2018). *Det omätbaras renässans: En uppgörelse med pedanternas världsherravälde* [*The renaissance of the unmeasurable: A settlement with the pedants' world domination*]. Volante.

Bostrom, N. (2014). *Superintelligence: Paths, dangers, strategies*. Oxford University Press.

Bower, J. L., & Christensen, C. M. (1995). Disruptive technologies: Catching the wave. *Harvard Business Review*, *73*(1), 43–53.

Bowker, G. C. (2013). Data flakes: An afterword to "raw data" is an oxymoron. In L. Gitelman (Ed.), *"Raw data" is an oxymoron* (pp. 167–171). MIT Press.

Bozdag, E. (2013). Bias in algorithmic filtering and personalization. *Ethics Information Technology*, *15*, 209–227. https://doi.org/10.1007/s10676-013-9321-6

Brin, S., & Page, L. (1998). *The anatomy of a large-scale hypertextual web search engine*. InfoLab, Stanford University. http://infolab.stanford.edu/~backrub/google.html

Brooks, F. (1975). *The mythical man-month: Essays on software engineering*. Addison-Wesley.

Bruno, G. (2018). *On magic* (Giordano Bruno collected works, Vol. 5). (S. Gosnell, Trans.). Windcastle Press. (Original essays published c. 1590–1592)

Bucher, T. (2017). The algorithmic imaginary: Exploring the ordinary affects of Facebook algorithms. *Information, Communication & Society*, *20*(1), 30–44. https://doi.org/10.1080/1369118X.2016.1154086

Bucher, T. (2018). *If… then: Algorithmic power and politics*. Oxford University Press. https://doi.org/10.1093/oso/9780190493028.001.0001

Bunz, M., & Meikle, G. (2018). *The Internet of things*. Polity Press.
Carrol, L. (1871). *Through the looking-glass, and what Alice found there.* McMillan.
Castells, M. (2000). *The rise of the network society: The information age* (2nd ed.). Blackwell.
CB Insights. (n.d.). The complete list of unicorn companies. Retrieved June 10, 2021, from https://www.cbinsights.com/research-unicorn-companies
Chandra, V. (2013). *Geek sublime: Writing fiction, coding software*. Faber & Faber.
Chang, E. (2018). *Brotopia: Breaking up the boys' club of Silicon Valley*. Portfolio.
Cheney-Lippold, J. (2017). *We are data: Algorithms and the making of our digital selves*. NYU Press. https://doi.org/10.2307/j.ctt1gk0941
Chilicorn Fund. (2018). *Projects*. Retrieved December 31, 2019, from https://spiceprogram.org/chilicorn-fund/
CHM. (2011). *Adele Goldberg: Bean bags and PARC culture*. Computer History Museum. https://www.computerhistory.org/revolution/input-output/14/348/2300
Chun, W. H. K. (2006). *Control and freedom: Power and paranoia in the age of fiber optics*. MIT Press. https://doi.org/10.7551/mitpress/2150.001.0001
Chun, W. H. K. (2008). On "sourcery," or code as fetish. *Configurations*, *16*(3), 299–324. https://doi.org/10.1353/con.0.0064
Chun, W. H. K. (2011). *Programmed visions: Software and memory*. MIT Press. https://doi.org/10.7551/mitpress/9780262015424.001.0001
Clark, L., Luske, H., Meador, J., & Reitherman, W. (Directors). (1959). *Donald in mathmagic land* [Film]. Walt Disney Prductions.
Comor, E., & Compton, J. R. (2015). Journalistic labour and technological fetishism. *The Political Economy of Communication*, *3*(2), 74–87. http://www.polecom.org/index.php/polecom/article/view/59
Computer Hope. (2017, October 17). *Wizard*. Retrieved December 2, 2019, from https://www.computerhope.com/jargon/w/wizard.htm
Conger, K. (2018, May 18). *Google removes 'don't be evil' clause from its code of conduct*. Gizmodo. https://gizmodo.com/google-removes-nearly-all-mentions-of-dont-be-evil-from-1826153393
Croll, A. (2012, August 2). *Big data is our generation's civil rights issue and we don't know it*. O'Reilly Radar. https://www.cc.gatech.edu/~beki/cs4001/big-data.pdf
Csikszentmihalyi, M. (1990). *Flow: The psychology of optimal experience*. Harper & Row.
Curran, J. P., & Gurevitch, M. (2005). *Mass media and society* (4th ed.). Arnold.
Cukier, K., & Mayer-Schoenberger, V. (2013). The rise of big data: How it's changing the way we think about the world. *Foreign Affairs*, *92*(3), 28–40. https://doi.org/10.1515/9781400865307-003
Dadson, S. A. (2016, November 11). *10 tech billionaire college dropouts*. Medium. https://medium.com/techtoday/10-tech-billionaire-college-dropouts-79b6c3a38afb
Dahlgren, P. (1996). Media logic in cyberspace: Repositioning journalism and its publics. *Javnost – the Public*, *3*(3), 59–72. https://doi.org/10.1080/13183222.1996.11008632
Dahlgren, P. (2009). *Media and political engagement: Citizens, communication, and democracy*. Cambridge University Press.
Dancy, C. (2018). *Don't unplug: How technology saved my life and can save yours too*. St. Martin's Press.
Darrah, C. N. (2008). Techno-missionaries doing good at the center. *Anthropology of Work Review*, *22*(1), 4–7. https://doi.org/10.1525/awr.2001.22.1.4
Dencik, L., Hintz, A., & Cable, J. (2016). Towards data justice? The ambiguity of anti-surveillance resistance in political activism. *Big Data & Society*, December, 1–12. https://doi.org/10.1177/2053951716679678
Descartes, R. (2017). *Meditations on first philosophy, with selections from the objections and replies* (2nd ed.). (J. Cottingham, Trans., Ed.). (Original work published in 1647). https://doi.org/10.1017/cbo9780511805028.006
Diakopoulus, N. (2016). Accountability in algorithmic decision making. *Communications of the ACM*, *59*(2). 56–62. https://doi.org/10.1145/2844110
Domingos, P. (2017). *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Penguin.
Dreyfus, H. L. (1972). *What computers can't do: Of artificial reason*. Harper & Row.

Duckworth, A., Peterson, C., Matthews, M. D., & Kelly, D. (2007). Grit: Perseverance and passion for long-term goals. *Journal of Personality and Social Psychology*, *92*(6), 1087–1101. https://doi.org/10.1037/0022-3514.92.6.1087

Duckworth, A. (2016). *Grit: Why passion and resilience are the secrets to success*. Vermillion.

Dyche, J. (2012, November 20). *Big Data "eurekas!" don't just happen*. Harvard Business review [Blog]. https://hbr.org/2012/11/eureka-doesnt-just-happen

Elias, N. (1998). The social constraint towards self-constraint. In S. Mennell, & J. Goudsblom (Eds.), *Norbert Elias on civilization, power and knowledge: Selected writings* (pp. 49–66). University of Chicago Press. (Original work written in 1939)

Ensmenger, N. (2012). *The computer boys take over: Computers, programmers and the politics of technical expertise*. MIT Press. https://doi.org/10.7551/*mitpress*/9780262050937.001.0001

Ensmenger, N. (2015). "Beards, sandals, and other signs of rugged individualism": Masculine culture within the computing professions. *Osiris*, *30*(1), 38–65. https://doi.org/10.1086/682955

Eubanks, V. (2017). *Automating inequality: How high-tech tools profile, police and punish the poor.* St. Martin's Press.

Evans-Pritchard, E. E. (1937). *Witchcraft, oracles and magic among the Azande*. Oxford University Press.

Firer-Blaess, S. (2016). *The collective identity of anonymous: Web of meanings in a digitally enabled movement* [Doctoral thesis, Uppsala University, Sweden]. http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-292734

Fisher, A. [Adam]. (2018). *Valley of genius: The uncensored history of Silicon Valley (as told by the hackers, founders, and freaks who made it boom)*. Hachette Book Group.

Fisher, A. [Alice]. (2017, October 15). Why the unicorn has become the emblem for our times. *The Guardian*. https://www.theguardian.com/society/2017/oct/15/return-of-the-unicorn-the-magical-beast-of-our-times

Fortea, J. A. (2014, April 3). *What is the difference between magic and religion?* Catholic Spiritual Direction. https://spiritualdirection.com/2014/04/03/difference-between-magic-religion

Frommer, D. (2014, September 11). *The hidden structure of the Apple keynote*. Quartz. https://qz.com/261181/the-hidden-structure-of-the-apple-keynote/

Frosh, P. (2019). You have been tagged: Magical incantations, digital incarnations and extended selves. In A. Lagerkvist (Ed.), *Digital existence: Ontology, ethics and transcendence in digital culture* (pp. 117–136). Routledge. https://doi.org/10.4324/9781315107479-6

Futurice. (2020). *Co-creating a resilient future: Our services*. Retrieved December 31, 2019, from https://futurice.com

Gardner, M. (1956). *Mathematics, magic and mystery*. Dover.

Gardner, M. (1957). *Fads and fallacies in the name of science*. Dover.

Giddens, A. (1984). *The constitution of society: Outline of the theory of structuration*. University of California Press.

Gibney, A. (Director). (2019). *The Inventor: Out for blood in Silicon Valley* [Film]. Jigsaw Productions; HBO Documentary Films.

Gibson, J. J. (1977). The theory of affordances. In R. Shaw, & J. Bransford (Eds.), *Perceiving, acting, and knowing: Toward an ecological psychology* (pp. 67–82). Erlbaum.

Gibson, W. (1984). *Neuromancer*. Penguin Random House.

Gillespie, T. (2018). *Custodians of the internet: Platforms, content moderation, and the hidden decisions that shape social media*. Yale University press

Girard, B. (2009). *The Google way: How one company is revolutionizing management as we know it*. No Starch Press.

Graham, P. (2010). *Hackers & painters: Big ideas from the computer age.* O'Reilly Media.

Harari, Y. N. (2014). *Sapiens: A brief history of humankind.* Harper.

Harari, Y. N. (2015). *Homo Deus: A brief history of tomorrow.* Vintage Books.

Haraway, D. (1985). A cyborg manifesto. *Socialist Review*. http://people.oregonstate.edu/~vanlondp/wgss320/articles/haraway-cyborg-manifesto.pdf

Harding, A., & Ho, J. (Directors). (2017). *Harry Potter: A history of magic* [Film]. Jude Ho; BBC.

Harrington, K. (2013, October 15). Is your company about to get Netflixed? *Forbes*. https://www.forbes.com/sites/kevinharrington/2013/10/15/is-your-company-about-to-get-netflixed/?sh=6880cd6e7499#537791317499

Harris, T. (2017). *How a handful of tech companies control billions of minds every day* [Video]. Ted conference. https://www.ted.com/talks/tristan_harris_how_a_handful_of_tech_companies_control_billions_of_minds_every_day

History Today. (2019). *Unicorns: A mythological creature of extraordinary resilience*. https://www.historytoday.com/archive/foundations/unicorns

Hjarvard, S. (2013). The mediatization of culture and society. Routledge. https://doi.org/10.4324/9780203155363

Hofstede, G. (1991). *Organisation och kulturer – om interkulturell förståelse* [*Organisation and cultures – about intercultural understanding*]. Studentlitteratur.

Hong, S.-H. (2019). Surveillance, sensors, and knowledge through the machine. In A. Lagerkvist (Ed.), *Digital existence: Ontology, ethics and transcendence in digital culture* (pp. 137–145). Routledge. https://doi.org/10.4324/9781315107479-7

Hylland Eriksen, T., & Nielsen, F. S. (2004). *Till världens ände och tillbaka. Socialantropologins historia* [*To the end of the world and back: A history of social anthropology*]. Studentlitteratur

Hägglund, M. (2019). *This life: Why mortality makes us free*. Profile Books.

IT & Telekomföretagen. (2019). *Statistik kvinnor och män* [*Statistics women and men*]. Retrieved April 7, 2020, from https://www.itot.se/om-oss/statistik/statistik-kvinnor-och-man/

Jacoby, J., Bourg, A., & Robertson, M. (Producers). (2020, February 18). Amazon empire: The rise and reign of Jeff Bezos (Season 38, Episode 14) [TV series episode]. *Frontline*. WGBH-TV.

Jarvis, J. (2009). *What would Google do?* Harper Luxe.

Jarzombek, M. (2016). *Digital Stockholm syndrome in the post-ontological age*. University of Minnesota Press.

Judge, M., Berg, A., Altschuler, J., Krinsky, D., Rotenberg, M., & Lassally, T. (Executive Producers). (2014–2019). *Silicon Valley* [TV series]. Judgemental Films; Altschuler Krinsky Works; Alec Berg Inc.; 3 Arts Entertainment; HBO Entertainment.

Juster, N. (1961). *The phantom tollbooth*. Random House.

Juster, N. (1996). *The phantom tollbooth* (Special 35th anniversary edition, with an appreciation by Maurice Sendak). Random House. (Original work published 1961)

Kaun, A., & Stiernstedt, F. (2020) Doing time the smart way? Temporality of the smart prison. *New Media & Society*, *22*(9), 1580–1599. https://doi.org/10.1177/1461444820914865

Kennedy, H. (2016). *Post, mine, repeat: Social media data mining becomes ordinary*. Palgrave Macmillan. https://doi.org/10.1057/978-1-137-35398-6

Kitchin, R. (2014). *The data revolution: Big Data, open data, data infrastructures & their consequences*. Sage.

Klein, H. K., & Kleinman, D. L. (2002). The social construction of technology: Structural consideratons. *Science, Technology, & Human Values*, *27*(1), 28–52. https://doi.org/10.1177/016224390202700102

Klein, E., Rozansky, K., Gordon, C., Mumm, C., Nishimura, L., Posner, J., Spingarn-Koff, J., & Townsend, K. (Executive Producers). (2018–present). *Explained* [TV series]. Vox.

Klinger, U., & Svensson, J. (2015). The emergence of network media logic in political communication: A theoretical approach. *New Media and Society*, *17*(8), 1241–1257. https://doi.org/10.1177/1461444814522952

Klinger, U., & Svensson, J. (2016). Network media logic: Some conceptual clarification. In A. Bruns, G. Enli, E. Skogerbö, A. Larsson, & C. Christensen (Eds.), *Routledge companion to social media and politics* (pp. 23–38). Routledge. https://doi.org/10.4324/9781315716299-3

Klinger, U., & Svensson, J. (2018). The end of media logics? On algorithms and agency. *New Media &* Society, *20*(12), 4653–4670. https://doi.org/10.1177/1461444818779750

Knuth, D. (1968). *The art of computer programming*. Addison-Wesley.

Kozinets, R. V. (2011). *Netnografi* [*Netnography*]. Studentlitteratur.

Kozulin, N. (2016, November 13). *Startup culture: Bean bag charis or something more?* Noa Kosulin's blog: Just another UBC blogs site [Blog]. https://blogs.ubc.ca/noakozulin/2016/11/13/startup-culture-bean-bag-chairs-or-something-more/

Kunda, G. (2006). *Engineering culture, control and commitment in a high-tech corporation*. Temple University Press.

Lagerkvist, A. (2017). Existential media: Toward a theorization of digital thrownness. *New Media & Society*, *19*(1), 96–110. https://doi.org/10.1177/1461444816649921

Lagerkvist, A. (2019) *Digital existence: Ontology, ethics and transcendence in digital culture*. Routledge. https://doi.org/10.4324/9781315107479

Larsson, S. (2005–2019). *Millennium* [Novel series]. Norstedts förlag.

Latour, B. (1993). *We have never been modern*. Harvard University Press

Lee, A. (2013, November 2). *Welcome to the unicorn club: Learning from billion-dollar startups*. TechCrunch [Blog]. https://techcrunch.com/2013/11/02/welcome-to-the-unicorn-club/

Levy, S. [Shawn]. (Director). (2013). *The internship* [Film]. Regency Enterprises; 21 Laps Entertainment; Wild West Picture Show Productions; TSG Entertainment.

Levy, S. [Steven]. (1984). *Hackers: Heroes of the computer revolution*. O'Reilly.

Levy, S. [Steven]. (2010). *Hackers: Heroes of the computer revolution* (with a new afterword). O'Reilly. (Original work published 1984)

Levy, S. [Steven]. (2012, February 3). Mark Zuckerberg, the hacker way and the art of the founder's letter. *Wired*. https://www.wired.com/2012/02/zuckerberg-hacker/

Lewis, S. C., & Usher, N. (2013). Open source and journalism: Toward new frameworks for imagining news innovation. *Media, Culture & Society*, *35*(5), 602–619. https://doi.org/10.1177/0163443713485494

Lexico. (2021a). *genius*. Retrieved February 27, 2020, from https://www.lexico.com/en/definition/genius

Lexico. (2021b). *nerd*. Retrieved January 2, 2020, from https://www.lexico.com/en/definition/nerd

Lorre, C., Prady, B., Aronsohn, L., Molaro, S., Kaplan, E., Ferrari, M., & Goetsch, D. (Executive Producers). (2007–2019). *The big bang theory* [TV series]. Chuck Lorre Productions; Warner Bros. Television.

Lowrey, W. (2017). The emergence and development of news fact-checking sites. Institutional logics and population ecology. *Journalism Studies*, *18*(3), 376–394. https://doi.org/10.1080/1461670X.2015.1052537

macessentials. (2009, January 24). *The lost 1984 video: Young Steve Jobs introduces the Macintosh* [Video]. YouTube. https://www.youtube.com/watch?v=2B-XwPjn9YY&list=PL1k7PwBgaszPZS0JkVU9M7SW7Hbo_mID9

Making the Macintosh. (n.d.). *The Xerox PARC visit*. https://web.stanford.edu/dept/SUL/sites/mac/parc.html

Malinowski, B. (1922). *Argonauts of the Western Pacific: An account of native enterprise and adventure in the archipelagoes of Melanesian New Guinea*. Routledge. https://doi.org/10.4324/9780203421260

Mansell, R. (2012). *Imagining the Internet: Communication, innovation and governance*. Oxford University Press.

MarcelVEVO. (2012, July 9). The mother of all demos, presented by Douglas Engelbart (1968) [Video]. YouTube. https://www.youtube.com/watch?v=yJDv-zdhzMY

March, J. G., & Olsen, J. P. (1984). The new institutionalism: Organizational factors in political life. *The American Political Science Review*, *78*(3), 734–749. https://doi.org/10.2307/1961840

Markoff, J. (2005). *What the dormouse said: How the sixties counterculture shaped the personal computer industry*. Penguin Random House.

Markoff, J. (2015). *Machines of loving grace: The quest for common ground between humans and robots*. Ecco Press.

McCrone, J. (1990). *The ape that spoke: Language and the evolution of the human mind*. William Morrow & Co.

McLuhan, M. (1964). *Understanding media*. McGraw-Hill.

Mead, M. (1928). *Coming of age in Samoa: A psychological study of primitive youth for Western civilization*. Morrow.

Metz, C. (2008, December 11). *The mother of all demos – 150 years ahead of its time*. The Register. https://www.theregister.com/2008/12/11/engelbart_celebration/

Morozov, E. (2011). *The net delusion: How not to liberate the world*. Penguin Books.

Morozov, E. (2013). *To save everything, click here: The folly of technological solutionism*. PublicAffairs.

Moore, P., & Robinson, A. (2015). The quantified self: What counts in the neoliberal workplace. *New Media & Society*, *18*(11), 2774–2792. https://doi.org/10.1177/1461444815604328

Mueller, J. P., & Massaron, L. (2017). *Algorithms for dummies*. John Wiley and Sons.

Nagy, P., Eschrich, J., & Finn, E. (2020). Time hacking: How technologies mediate time. *Information, Communication & Society*. Advance online publication. https://doi.org/10.1080/1369118X.2020.1758743

Neff, G., & Nafus, D. (2016). *Self-tracking*. MIT Press. https://doi.org/10.7551/mitpress/10421.001.0001

Noble, S. U. (2018). *Algorithms of oppression: How search engines reinforce racism*. NYU Press. https://doi.org/10.2307/j.ctt1pwt9w5

O'Neill, C. (2016). *Weapons of math destruction*. Crown Publishing.

Oppland, M. (2021, February 15). *8 ways to create flow according to Mihaly Csikszentmihalyi*. Positive Psychology. https://positivepsychology.com/mihaly-csikszentmihalyi-father-of-flow/

Oyserman, D., Coon, H. M., & Kemmelmeier, M. (2002). Rethinking individualism and collectivism: Evaluation of theoretical assumptions and meta-analyses. *Psychological Bulletin*, *128*(1), 3–72. https://doi.org/10.1037/0033-2909.128.1.3

Parks, L., & Starosielski, N. (2015). *Signal traffic: Critical studies of media infrastructures*. University of Illinois Press.

Pasquale, F. (2015). *The black box society: The secret algorithms that control money and information*. Harvard University Press.

Paul, K. (2020, February 3). 'They know us better than we know ourselves': How Amazon tracked my last two years of reading. *The Guardian*. https://www.theguardian.com/technology/2020/feb/03/amazon-kindle-data-reading-tracking-privacy

Paul Rey. (2019, March 8). *GOOD FOR ME [Live 35 000 ft in the air]* [Video]. YouTube. https://www.youtube.com/watch?v=qY53m63NQKU

Peters, J. D. (2015). *The marvelous clouds: Toward a philosophy of elemental media*. University of Chicago Press. https://doi.org/10.7208/chicago/9780226253978.001.0001

Prensky, M. (2009). H. Sapiens digital: From digital immigrants and digital natives to digital wisdom. *Innovate: Journal of Online Education*, *5*(3), Article 1. https://nsuworks.nova.edu/innovate/vol5/iss3/1

Prometheus. (2021). *Overview: What is Prometheus?* Retrieved January 1, 2020, from https://prometheus.io/docs/introduction/overview/

Pyper, B. (n.d.). *A unicorn taco holder exists to make eating pure magic*. Totally the bomb [Blog]. https://totallythebomb.com

Raley, R. (2013). Dataveillance and countervailance. In L. Gitelman (Ed.), "*Raw data" is an oxymoron* (pp. 121–145). MIT Press.

Roberts, S. T. (2019). Behind the screen: Content moderation in the shadows of social media. Yale University Press. https://doi.org/10.2307/j.ctvhrcz0v

Robinson, A. (2018, March 12). Want to boost your bottom line? Encourage your employees to work on side projects: Three ways you can run side projects at your company like Google does. *Inc*. https://www.inc.com/adam-robinson/google-employees-dedicate-20-percent-of-their-time-to-side-projects-heres-how-it-works.html

Rosales, A., & Svensson, L. (2021). Perceptions of age in contemporary tech. *Nordicom Review*, *41*(1), 79–92. https://doi.org/10.2478/nor-2021-0021

Rosenberg, S. (2008). *Dreaming in code: Two dozen programmers, three years, 4,732 bugs, and one quest for transcendent software*. Crown Publishing.

Rosenberg, D. (2013). Data before the fact. In L. Gitelman (Ed.), "*Raw data" is an oxymoron* (pp. 15–40). MIT Press.

Rotenberg, J. (2014, January 27). *The very first 'Stevenote'* [CHM Blog]. Computer History Museum. https://computerhistory.org/blog/the-very-first-stevenote/?key=the-very-first-stevenote

Rushkoff, D. (2019). *Team human*. W. W. Norton.

Russell, S. (Developer). (1962). *Spacewar!* [Videogame]. PDP-1.

Ruth, A. (2016, August 22). *Thomas Edison – 10,000 ways that won't work*. Due [Blog]. Retrieved January 8, 2020, from https://due.com/blog/thomas-edison-10000-ways-that-wont-work/

Sandvig, C., Hamilton, K., Karahalios, K., & Langbort, C. (2016). When the algorithm itself is a racist: Diagnosing ethical harm in the basic components of software. *International Journal of Communication*, *10*, 4972–4990. https://ijoc.org/index.php/ijoc/article/view/6182

SCB. (2017). *Fler män än kvinnor programmerar* [*More men than women program*]. Retrieved April 7, 2020, from https://www.scb.se/hitta-statistik/statistik-efter-amne/levnadsforhallanden/levnadsforhallanden/befolkningens-it-anvandning/pong/statistiknyhet/it-bland-individer-2017/

Seaver, N. (2013). Knowing algorithms. *Media in Transition 8*. http://nickseaver.net/papers/seaverMiT8.pdf

Stack Exchange. (2021). *Origin of the term "wizard" in computing*. English language & usage. Retrieved December 3, 2019, from https://english.stackexchange.com/questions/65728/origin-of-the-term-wizard-in-computing

Steadman, I. (2013, January 25). Big data and the death of the theorist. *Wired Magazine*. https://www.wired.co.uk/article/big-data-end-of-theory

Steiner, C. (2012). *Automate this: How algorithms came to rule our world*. Penguin Books.

Stevenson, A. (2015, September 23). *World's youngest female billionaire – next Steve Jobs?* CNBC. https://www.cnbc.com/2015/09/23/worlds-youngest-female-billionaire-next-steve-jobs.html

Stross, R. (2008). *The wizard of Menlo Park: How Thomas Alva Edison invented the modern world*. Random House.

Stöppler, M. C. (Ed.). (2021). *Medical definition of apophenia*. Medicine Net. Retrieved June 11, 2021, from https://www.medicinenet.com/apophenia/definition.htm

Svensson, J. (2011). The expressive turn of citizenship in digital late modernity. *JeDEM eJournal of eDemocracy*, *3*(1), 42–56. https://doi.org/10.29379/jedem.v3i1.48

Svensson, J. (2012). Social media and the disciplining of visibility: Activist participation and relations of power in network societies. *European Journal of E-Practice*, *16*, 16–28. http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-206226

Svensson, J. (2015). Political participation on social media platforms in Sweden today: Connective individualism, expressive issue-engagement and disciplined updating. International Journal of Media & Cultural Politics, 10(3), 347–354. https://doi.org/10.1386/macp.10.3.347_3

Svensson J. (in review). *Behind the news-ranking algorithm: Actors, conflicts and logics when introducing algorithmic automation*.

Svensson, J., & Poveda Guillén, O. (2020). What is data and will it make humans obsolete? Key questions in the age of data-essentalism. *Journal of Digital Social Research*, *2*(3), 65–83. https://doi.org/10.33621/jdsr.v2i3.40

Svensson, J., & Wicander, G. (Eds.). (2010, November 10–11). *Proceedings of the 2nd international conference on M4D mobile communication technology for development, Kampala, Uganda*. Centre for HumanIT, Karlstad University, Sweden. http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-6480

Sveriges Radio. (2019, November 15). Det expanderade livet, reglera regelverket & från hat till humor [The expanded life, regulating regulations & from hate to humor]. https://sverigesradio.se/avsnitt/1389370

Sveriges Radio. (2020a, March 29). *Krönikör Helena Granström: Naturen är en del av oss* [*Columnist Helena Granström: Nature is a part of us*]. https://sverigesradio.se/artikel/7440725

Sveriges Radio. (2020b, May 24). *Gud som förstelnad metafor* [*God as a frozen metaphor*]. https://sverigesradio.se/artikel/7480190

Syvertsen, T., & Enli, G. (2019). Digital detox: Media resistance and the promise of authenticity. *Convergence*, *26*(5-6), 1269–1283. https://doi.org/10.1177/1354856519847325

TEDx Talks. (2015, February 5). *Technology contributions by underrepresented minorities | James West* [Video]. YouTube. https://www.youtube.com/watch?v=FHSTU1zLFJA

Tegmark, M. (2017). *Life 3.0: Being human in the age of artificial intelligence*. Vintage Books.

Tellonym. (n.d.). *Info center*. Retrieved November 15, 2019, from https://tellonym.me

Tenhaven, J. (Writer & Director). (2017). *The Silicon Valley revolution – How a bunch of nerds changed our lives* [TV series]. Arte; ECO Media TV; Westdeutscher Rundfunk.

The Economist. (2017, May 6). The world's most valuable resource is no longer oil, but data. *The Economist*. https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data

The Retronaut. (2014, September 25). *1984: Ridley Scott's first Apple Macintosh commercial* [Video]. YouTube. https://www.youtube.com/watch?v=0q7iX0QWaTg

Thiessen, J., Wootton, J., Miller, J., Lu, D., Stuby, T., & Myhre, M. (Directors). (2010–2019). *My little pony: Friendship is magic* [TV series]. Allspark Animation; Studio B Productions; DHX Studios Vancouver.

Thomas, D. (2002). *Hacker culture*. University of Minnesota Press.

Time. (1995, March 1). *Cyberspace*. http://content.time.com/time/covers/0,16641,19950301,00.html

Tufekci, Z. (2015). Algorithmic harms beyond Facebook and Google: Emergent challenges of computational agency. *Journal on Telecommunication & High Tech*, *13*, 203–218.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *49*, 433–460. https://doi.org/10.1093/mind/LIX.236.433

Turner, F. (2006). *From counterculture to cyberculture: Stewart Brand, the Whole Earth network and the rise of digital utopianism*. University of Chicago Press.

Turner, F. (2009). Burning Man at Google: A cultural infrastructure for new media production. *New Media & Society*, *11*(1-2), 73–94. https://doi.org/10.1177/1461444808099575

Turkle, S. (1995). *Life on the screen: Identity in the age of the Internet*. Simon & Schuster.

Turkle, S. (2011). *Alone together: Why we expect more from technology and less from each other*. Basic Books.

Törnberg, A. (2019). Teorins död? Om framväxten av en digital empirism [The death of theory? On the emergence of a digital empiricism]. *Fronesis*, (64-65), 132–146.

van Dijk, J. (2006). *The network society* (2nd ed.). Sage.

van Dijck, J. (2013). *The culture of connectivity: A critical history of social media*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199970773.001.0001

van Dijck, J. (2014). Datafication, dataism and dataveillance: Big data between scientific paradigm and ideology. *Surveillance and Society*, *12*(2), 197–208. https://doi.org/10.24908/ss.v12i2.4776

Visual Paradigm. (2020). *What is agile software development?* https://www.visual-paradigm.com/scrum/what-is-agile-software-development/

Wachter-Boettcher, S. (2017) *Technically wrong: Sexist apps, biased algorithms, and other threats of toxic tech*. W. W. Norton.

Walliams, D., & Lucas, M. (Writers & Creators). (2003–2007). *Little Britain* [TV series]. BBC.

Wareham, J. (2018, August 17). *Unicorns are the queer icons of our time*. Gay Star News. https://www.gaystarnews.com/article/evidence-unicorns-are-queer-icons/

Weizenbaum, J. (1976). *Computer power and human reason: From judgement to calculation*. W. H. Freeman & Company.

Weizenbaum, J., & Wendt, G. (2015). *Islands in the cyberstream: Seeking havens of reason in a programmed society*. Litwin Books.

Whittaker, X. (2018). There is only one thing in life worse than being watched and that is not being watched: Digital data analytics and the reorganization of newspaper production. In P. Moore, M. Upchurch, & X. Whittaker (Eds.), *Humans and machines at work: Dynamics of virtual work* (pp. 73–99). Palgrave Macmillan. https://doi.org/10.1007/978-3-319-58232-0_4

Wiener, N. (1948). *Cybernetics, or control and communication in the animal and the machine*. MIT Press. https://doi.org/10.7551/mitpress/11810.001.0001

Wikipedia. (2021a). *Grit (personality trait)*. Retrieved January 8, 2020, from https://en.wikipedia.org/wiki/Grit_(personality_trait)

Wikipedia. (2021b). *Mariner 1*. Retrieved April 7, 2020, from https://en.wikipedia.org/wiki/Mariner_1

Wikipedia. (2021c). *Mathemagician*. Retrieved March 4, 2020, from https://en.wikipedia.org/wiki/Mathemagician#cite_note-mathem-1

Wikipedia. (2021d). *Prometheus*. Retrieved January 1, 2020, from https://en.wikipedia.org/wiki/Prometheus

Wikipedia. (2021e). *Stevenote*. Retrieved January 9, 2020, from https://en.wikipedia.org/wiki/Stevenote

Wikipedia. (2021f). *Wizard (software)*. Retrieved December 3, 2019, from https://en.wikipedia.org/wiki/Wizard_(software)

Wizard Amigos. (n.d.). *Open source JavaScript e-learning for cyber nomads*. Retrieved December 2, 2019, from https://wizardamigos.com

Wood, J. (2018). *Fantastic creatures in mythology and folklore: From medieval times to the present day*. Continuum. https://doi.org/10.5040/9781474204446

World History Encyclopedia. (2013, April 20). *Prometheus*. https://www.worldhistory.org/Prometheus/

Yarow, J. (2015, April 4). *Silicon Valley is 'incredibly white and male' and there's a 'sort of pride' about that fact, says Silicon Valley culture reporter*. Business Insider. https://www.businessinsider.com/silicon-valley-is-incredibly-white-and-male-2015-4/lightbox?r=AU&IR=T

Zuboff, S. (2019). *The age of surveillance capitalism*: *The fight for a human future at the new frontier of power*. Profile Books.

Zuckererg, M. (2012, February 1). Letter from Mark Zuckerberg. In Facebook, Inc., *Form S-1 registration statement under the securities act of 1933* (pp. 67–70). https://www.sec.gov/Archives/edgar/data/1326801/000119312512034517/d287954ds1.htm#toc287954_10

In our connected data societies, the importance of algorithms and automated systems is obvious. They determine search engines' rankings, what driverless cars do when a child appears on the road, and stock market changes. Today data-driven algorithms and automated systems are everywhere.

While algorithms and automated systems themselves are often a topic of controversy and debate, this book is about the people behind them; it is an account of the cultures, values, and imaginations that guide programmers in their work designing and engineering software and digital technology. Technology, it is argued, is not neutral and developed free of context. And since algorithms and automated systems exercise power in connected data societies, it is pivotal to understand their creators, who could be labelled, it is argued in the book, Wizards of the Web.

This book is the result of an ethnographically inspired study based on interviews with software engineers and programmers, observations made at tech headquarters and conferences in Denmark, Sweden, Brazil, Germany, India, and the US, and a case study of the introduction of algorithmic automation on the front page of a Scandinavian newspaper.

The author, Jakob Svensson, is professor of Media and Communication Studies at Malmö University. The book is part of the research project Behind the Algorithm (funded by the Swedish Research Council, 2018–2020).

NORDICOM

UNIVERSITY OF GOTHENBURG

Nordic Council of Ministers

Nordicom is a centre for Nordic media research at the University of Gothenburg, supported by the Nordic Council of Ministers.

ISBN 978-91-88855-52-7

Nordicom, University of Gothenburg, PO Box 713, SE-405 30 Gothenburg, Sweden
www.nordicom.gu.se