Value-Sensitive Software Design

Paulan Korenhof

Contents

W. Liebregts et al. (eds.), *Data Science for Entrepreneurship*, Classroom Companion: Business,
https://doi.org/10.1007/978-3-031-19554-9_21

21

Learning Objectives

- Understand why technology is not a neutral instrument
- Be able to recognise the non-neutral impact of a particular technology
- Think about how to embed ethical values in software design

21.1 Introduction

Software is involved in almost everything in our daily lives: it is in our desktops, laptops, and smartphones, but we see it also implemented in an increasing range of common-use items like cars, bikes, electric toothbrushes, cookers, fitness devices, and toys. We use software to pay, communicate, shop online, plan a route, plan public transport, watch series and movies, decide which insurance to take, which political party to vote for, order food, check our health, and so on. And it is not just us as private citizens that so gladly use software for many things in our daily practice. Governmental institutions and business rely heavily on software for performing many of their processes. They at times even fully automate certain decision-making processes, like deciding whether someone should be given a fine for speeding, whether someone should be granted a loan or a credit card, or whether someone is a promising job applicant.

While there are many advantages to using software as an instrument to help us out with all sorts of tasks, there is a catch. The catch with software programs is that like all technology, they are inherently *not neutral*: every technology has a certain bias, a particular way in which it likely affects our practices, our choices, our perception, and how we interpret the world around us. Meanwhile, the impact of software on the lives of individuals can be high, especially as increasingly more elements of our lives are dependent on and intertwined with these applications.

The goal of this chapter is to draw the readers' attention to this non-neutral character of technology and to encourage them to try to utilise this non-neutrality in a beneficial manner. The chapter starts with discussing why technology is never a neutral instrument. With the help of various examples relating to software, the impact of technology on different elements of human life is discussed. Given the inherently non-neutral character of technology and its potential problematic impact, it is important to figure out how to reap the fruits of the technology while reducing its potential harms. The second half of the chapter therefore argues that, ideally, we should actively deal with this non-neutrality from the very beginning of technology design. In order to help designers with this, the chapter introduces the main ideas underlying value-sensitive design (VSD). Because it is not possible to provide here a complete instruction manual for value-sensitive software design, the chapter aims to give readers sufficient food for thought so that they can venture on the follow-up journey themselves.

21.2 The Good, the Bad, and the Never Neutral

In this section, we will delve into the non-neutrality of technology. First, the background of the non-neutrality perspective will be discussed. After that, the non-neutrality will be explained in more detail by approaching the technology from a micro- and a macro-level perspective.

21.2.1 Non-neutrality

The importance of technology for human life can hardly be overestimated: society and life as we know it today would not exist without the development and use of technology. Technology allows us to achieve certain goals, do and perceive things we could not do without the use of technology, and reveal the world in new ways to us. For example, we can see single cells of the body through a microscope, consult people at the other side of the planet over the phone, or look into the body with an echo device. By allowing us such new experiences, actions, and perceptions of the world, technology enables us to relate in the world in new manners and affects our interpretation of the world around us, as well as our practices and social conventions (Kiran & Verbeek, 2010; Verbeek, 2011). Due to the shaping influence of technology on our perception, experience, actions, goals, and understanding, technology transcends the role of being merely an instrument. In the last century, philosophers of technology therefore argued that technology is inherently *not neutral*: technologies can reveal the world to us in new ways; create new choices and possibilities for action; establish social identities, power relations, and occasions of inclusion and exclusion; and influence and inform us, our choices, our culture, and our world views (see, e.g., Heidegger, 1954; Ihde, 1983; Latour, 1993; Feenberg, 2002; Verbeek, 2005). Due to this non-neutrality, technology has a normative impact on the relation between human beings and their world (Hildebrandt, 2015).

The analysis of the impact and meaning of technology for human existence gave rise to different schools of thought in the philosophy of technology. Instead of dragging the reader into the discussion between the various ideas, it will be more valuable for the purpose of this chapter to take the two main directions of the perspectives into account, albeit in a simplified manner, and understand them as complementary to each other. Simplified, we can say that technology affects the way humans engage with the world that occurs stretching from a micro-level, an individual and empirical level (see, e.g., Ihde, 1983; Verbeek, 2005), to macro-level, a societal and abstract level (see, e.g., Stiegler, 1998; Feenberg, 2002). While taking the technology's impact on a more macro-level into account is indispensable for getting an understanding of the scope and depth of its impact on our lives, an analysis focused more on a micro-level can be very helpful to trace some of the problems back to particular concrete properties of the technology. However, I argue that the micro- and macro-level of the impact cannot be fully separated from each other because the micro-mechanics give shape to the impact on the macro-level, while what happens at the macro-level is bound to influence how the micro-mechanics are realised in practice. Despite this, I will for structural clarity start with a focus on the micro-level and from thereon move to a more macro-level of impact—but readers should note that these are linked.

21.2.2 Impact on a Micro-level

On a micro-level, technology affects the human perception, actions, practices, and goals, by letting us experience the world *mediated* by the technology (see, e.g., Ihde, 1983; Verbeek, 2005). For example, a thermometer can show us the temperature of our bodies. If the thermometer indicates 38.5 °C, we will likely conclude that we have a fever, even if we do not feel ill. By telling us that we are in fact ill, while we may feel fine, the technology affects how we understand our health. By doing so, the technology co-shapes our relation to the world (in this case, the human body). Technology affects our perception, our actions, and even how we think and what we remember. The latter was clearly shown by research into the effects of search engines on our memory: it turned out that once people know that they can rely on a search engine or the like for their information, they tend to remember where and how to find something instead of remembering the content itself that they needed to recall (Sparrow et al., 2011, p. 778).

When mediating our relation to the world, technology generally has a particular focus: it often reveals and highlights particular aspects of a technological co-shaped reality, while other elements are obscured or ignored (Verbeek, 2005, p. 131). Think for instance about what happens when you make a telephone call. When you make a telephone call, this makes the voice of the caller stand out, while the rest of the individual is concealed. The technology hereby establishes a particular relation between a human being and the world, a relation that is directed towards something (i.e. in the case of the telephone call, the technology is directed towards sound). We can therefore say that technology has a certain "directionality" (Verbeek, 2005, p. 115). This directionality is embedded in the material design of the technology. With its directionality, the technology takes a certain "stance": it can "*suggest, enable, solicit, prompt, encourage, and prohibit certain actions, thoughts, and affects or promote others*" (Lazzarato & Jordan, 2014, p. 30). A designer will generally aim to give the technology a particular directionality by imbuing the technology with certain properties. For example, web applications of online stores and services are generally designed in such a manner that they render it impossible for users to place an online order for a product or service without accepting the company's general terms and conditions. The directionality of the technology in this case is designed so that it ensures the legal base and protection of the company selling the product or service. However, there is a limit to the influence of the designer: in the end is the technology itself that is present in the world and with which users interact. Technology has a certain autonomous existence separate from its designers (Chabot, 2013, p. 15). As such, technology can also easily have effects or uses that are unforeseen or unintended by its designers. However, as the designers determine the material properties of the technology, they do play a pivotal role in the shaping of the non-neutral directionality of the technology they design.

The directionality of software thus depends on the characteristics and constraints of the technology itself, as well as on the choices made by its designer. The developer's knowledge, expertise, cultural background, and limitations form the

background of the software design (Kitchin, 2017, p. 18). This takes shape in a two-step process: a developer needs (1) to interpret the task at hand and (2) translate this into code. As designers develop the software, their views on how to interpret and translate certain concepts, values, and goals form the base on which they encode procedures that determine what the software does and does not do. With this, the designer's assumptions and biases are incorporated in the software—whether it be intentional or accidental (Friedman & Nissenbaum, 1996; Goldman, 2008). Coded procedures are therefore inevitably value laden (see, e.g., Brey & Søraker, 2009; Mittelstadt et al., 2016).

► Example 1

Imagine that a company hires you to circulate a vacancy for a truck driver and select some potential candidates. You decide to develop an online form for the job application procedure. In order to apply, job applicants need to fill in their name and date of birth, upload a CV, tick a box with either male or female, and tick a box that they consent to the processing of their personal data. In order to prevent people from forgetting to add anything, you make all the fields mandatory in order to apply. While such a form may seem simple and straightforward, in this small application, we can already see many points on which the software has a certain directionality towards the world—and may even be problematic. First off, the fact that it is an online form immediately places the application process in the digital realm. As such, the application may bypass the less digital skilled or people who have limited access to the internet. Secondly, as the form needs to be fully completed before someone can apply, the software compels people to reveal all this information (or lie about it) and to identify themselves according to the options the form offers. The form expresses a certain view of the world: it presents certain elements as important about people who want to be a truck driver and expresses a binary gender perspective. Job applicants can experience this "identity fit to the software's box" as problematic: people may prefer not to give their date of birth for privacy reasons or as protection against age discrimination, people may not identify themselves as male or female or find it irrelevant to share in a job application, some people may prefer more creative freedom when applying for a job, etc. However, the choices people can make in applying for the vacancy are limited to and determined by the options offered by the online form. ◄

Giving shape to software can be especially tricky when designing software that needs to produce decisions based on particular laws. Examples are the automatic fines issued when someone is recorded by a traffic camera for speeding, or the allocation of child care subsidy based on a combination of data sets. In such cases, programmer will need to translate law or policy rules into code and may find themselves confronted with questions about when *exactly* a certain case falls under the definitions of a particular law or policy rule. When giving shape to such boundaries in code, the programmer fills in legal concepts and de facto establishes a policy rule by programming the software.

The role of design choices in software can hardly be overestimated: the code is what the software does and controls what users can and cannot do. This is what Lessig meant with his famous statement "code is law" (Lessig, 2006). However, as

Pariser explains, software code more forcefully controls user behaviour than law, at least at the outset:

21

» "If code is law, software engineers and geeks are the ones who get to write it. And it's a funny kind of law, created without any judicial system or legislators and enforced nearly perfectly and instantly. Even with antivandalism laws on the books, in the physical world you can still throw a rock through the window of a store you don't like. You might even get away with it. But if vandalism isn't part of the design of an online world, it's simply impossible. Try to throw a rock through a virtual storefront, and you just get an error" (Pariser, 2011, pp. 96–97).

The developer thus exercises a significant degree of power over users through the software's architecture: users can only use the software as its architecture allows. In this, the software's interface plays a key role on the one hand by suggesting to the user what software does and can do, while on the other hand it is also the realm of interaction by means of which users can operate the software. The interface thus shapes the perception of users, provides them with the know-how of the software, and offers them a particular set of actions—and without using tricks, the user's perception and actions are commonly limited to that which is offered by the interface. Commonly, this is a graphic user interface (GUI) that hides the source code of the software and thereby often renders a substantial part of what the software actually does opaque. Moreover, the interface can be shaped to manipulate users to perform certain actions by means of persuasion or nudging (see, e.g., Fogg, 1999; Harjumaa & Oinas-Kukkonen, 2007; Thaler & Sunstein, 2009). Think for example of the various ways in which website designers try to nudge users into accepting marketing cookies by using a big green button to "accept all cookies", versus a less visible small red button that allows a user to select a different setting (see the chapter by Gellert in this book for the explanation of consent and the use of cookies).

Depending on the interface, users are thus offered a more or lesser degree of insight into the operations performed by the software and are given certain choices with regard to the actions that they can perform. This affects the autonomy of users: their ability to self-govern, which entails the freedom to make informed decisions and shape their lives as they see fit.[1] For example, some online stores require users to classify themselves as "female" or "male" before they can place an order; other stores give them more options, like the extra option of "I'd rather not say"; while again other stores do not mark gender as a required field at all and leave it to the users to decide whether they want to fill in the field. The more freely users can choose and act, the more autonomy they have. Reducing the autonomy of users and forcing them down certain action paths can estrange human agents from the task they are performing with the software, especially if they have little insight into and know-how of what the software is actually doing behind the interface. De Mul and Van den Berg therefore argue: "Awareness of, and insight into the 'scriptal

1 The exact definition of what autonomy entails differs somewhat per social and political perspective. For the purposes of this chapter, I kept the concept relatively open and phrased it in a manner that can give some handholds in relation to software design.

character' of the artefact, and having the ability to influence that character, is crucial for users in the light of the delegation of their autonomy" (De Mul & van den Berg, 2011, pp. 59–60).

21.2.3 Impact on a Macro-level

Technology not only affects processes, practices, and perception on an individual level, but also influences our lives on a macro-level: it influences and even shapes societal organisation and transactions, institutions, governmental agency, politics, science, relations between individuals, and even our identity (Stiegler, 1998). Especially on the level of the functioning of companies and institutions, as well as the work of people therein, the use of software deeply affects the processes and output, which in turn can affect people outside of the organisation), and even society at large. Take for instance the use of automated decision-making software, like the automated issuing of fines for speeding. This entails a shift in decisionary power, whereby the main "decision maker" changes from a human agent who learned to employ their knowledge of legal rules and policy in order to make a contextual assessment, to software that strictly applies rules:

> » decisions are pre-programmed in the algorithms that apply the same measures and rules regardless of the person or the context (e.g., a speeding camera does not care about the context). Responsibility for decisions made, in these cases, has moved from 'street-level bureaucrats' to the 'system-level bureaucrats', such as managers and computer experts, that decide on how to convert policy and legal frameworks into algorithms and decision-trees (Noorman, 2020).

By shifting the decisionary power, such software generally reduces the space for individual discretion and gives rise to a workforce that mass produces decisions in a uniform production process on which they have little influence (Giritli Nygren, 2009; Wihlborg et al., 2016). As such, software "reframes relationships, responsibilities and competences" (Wihlborg et al., 2016, p. 2903).

Moreover, when know-how is embedded into software, the practical need for human agents to have this same know-how reduces and sometimes even disappears: a click on a button can be enough to provide users with what they need. An example of this is a bank where "customer advisers get predetermined interest rates from the IT system for their customers' credit, but they do not know how this interest rate is calculated or what justifies it" (Spiekermann, 2015, p. 12). With the delegation of know-how to software, human agents, and society in general, are becoming increasingly dependent on software for many of their processes. Stiegler therefore argues that technology is in a sense a poison that is at the same time its own cure—a *pharmakon* (Stiegler, 2012): while software allows humans to forget knowledge and how to do certain things (poison), it at the same time remedies this loss of know-how by performing the actions for them (cure). Think for instance of phone numbers. In the pre-mobile phone era, the phone numbers were not stored in the telephone. This commonly meant that you automatically memorised the numbers of family and close friends because you had to consistently type the num-

ber in and, also, putting in some effort to remember a number was faster than having to look the number up in an address book. However, with smartphones, this need for remembering became virtually superfluous and typing in the number is unnecessary: the technology does this for us. The result is that we are far less likely to remember phone numbers unless we actively spend effort to memorise them. The effect of this becomes painstakingly clear when you lose access to the contact list in your phone.

The more dependent we become on particular software, the more power it holds over us. We can see this clearly in the use of search engines. Due to the abundance of information resources on the web, we have become highly dependent on their use to find online information. As such, this pivotal position of search engines imbues them with a significant power over the connection between users and content providers: search engines "are attention lenses; they bring the online world into focus. They can redirect, reveal, magnify, and distort. They have immense power to help and hide" (Grimmelmann, 2010, p. 435). Dropping out of a search engine's search result list can render content nearly invisible to a significant part of the web users—with all due consequences for the web content owner as well as for searching users.

The power of software is strengthened by the trust people tend to have in the technology to fulfil its tasks properly: people tend to have an "automation bias" due to which they trust the output of software more than their own assessment (see, e.g., Skitka et al., 1999). As such, they may overly rely on software for their assessment or to quickly make decisions (Skitka et al., 2000). Combine the human inclination towards automation bias with an opaque interface that suggests an objectivity or neutrality of the software's operations, while the software is in fact bound to harbour some (intentional or unintentional) biases and maybe even has some errors, and we have a recipe for disaster.

The scale of processing afforded by software can magnify the impact of its biases to a society-wide level. Software may "normalize the far more massive impacts of system-level biases and blind spots" (Gandy, 2010, p. 33). Take for example social media. This type of software led to changes in web culture by giving rise to new standards of what is considered "normal" (Van Dijck, 2013; Wittkower, 2014). One of the changes brought about by social media is a shift from relatively anonymous online communication to pattern communication where "individuals are increasingly known, and in fact willingly share a lot of their personal information online" (Sparrow et al., 2005, p. 283). A pivotal role here is played by the software's default settings. The default settings set a standard for its use and require users with divergent preferences to invest time and effort in order to adjust the default (see, e.g., van den Berg & Leenes, 2013; Acquisti et al., 2015). The default settings thereby express a certain world view, a "normalcy", with regard to its use. For example, originally on Facebook, the default setting of an account was that all user information and posts were publicly available. With these default settings, Facebook suggested that the standard was to be available, accessible, and identifi-

able as a particular offline person for a potentially worldwide audience.[2] Additionally, users tend to have an inclination to accept the default settings, because it "is convenient, and people often interpret default settings as implicit recommendations" (Acquisti et al., 2015, p. 512). The default settings thus strongly affect user behaviour and norms.

Last, the output of software can impact the lives of people who are not the software's users—as well as society at large. For instance by placing certain groups of people at a disadvantage. Let us look a bit deeper into this by focusing on automated decision-making applications like those that issue fines for speeding, mark people as being fraud risks, calculate the amount people need to pay for their insurance policy, etc. In these cases, people who are not the initial users of the software are subjected to the output (decision) produced by the software. However, as the transparency of the software's output is dependent on what is programmed *into* the software to show as output, people may be profiled and subjected to a decision of which the how, why, and what are not made clear to them. As such, it is difficult for them to figure out if an error was made, and if so, where and how. A lack of insight in what happens in software can be particularly problematic in the case of automated decision-making software used by government agencies because these agencies have the obligation to be transparent in their decisions and motivate them. Moreover, the lack of transparency and access to the same software makes it difficult for people to effectively challenge an automatically produced decision. It leads to a power imbalance by establishing an inequality of arms between a common citizen and the agency—which generally already holds a power position because people are dependent on the agency for one thing or the other. These are only some of the issues concerning automated decision-making software. The impact of automated decision-making software on our lives and world, a full discussion here is too extensive.

21.2.4 In Sum

This section discussed why software, like all technologies, is not a neutral instrument. Software has a certain directionality in which it likely affects and even changes the manner in which we work, decide, and interact. Its impact can stretch deep into society and especially into the lives of people. The question now is how should we deal with this non-neutrality.

2 Under pressure of public institutions and European legislation, Facebook eventually changed its default settings to a restricted audience and with that sets a somewhat more privacy-friendly standard.

21.3 Employing the Never Neutral

This section offers an approach on how to deal with the non-neutrality of technology. It will start by arguing for a proactive approach with regard to values in technology design. In order to give some handholds on how to start, the section then gives a general outline of "value-sensitive design". Last, given the focus of this chapter on software, this section takes a look at the values promoted by the General Data Protection Regulation (GDPR) when it comes to the processing of personal data.

21.3.1 A Challenge for Designers

While technology is not necessarily good or bad, it is thus never neutral. The design of technology is therefore pivotal: at this stage of the process, a significant part of what a specific technology does and does not do, its directionality, is determined. De Mul and Van den Berg therefore point out that, despite the strong influence of technology on us and our world, "the *responsibility* for that world and what happens in it is still in the hands of human beings and not in the hands of the technologies. After all, human beings are the architects, designers and users of the technologies, and for that reason they are responsible for their creations and their creations' output" (De Mul & van den Berg, 2011, p. 46). Technology is designed by us, and in many cases, we will be able to design the technology in such a manner that we can reduce, or even prevent, its problematic impact.

A way to deal with the inherent non-neutrality of technology is therefore to consciously *design* technology in a manner that it supports or promotes certain social or moral values, like freedom, safety, and privacy. Already in the design process, we should therefore be asking what the potential impact of a software application may be, and how the application should work if we want it to promote certain values, while repressing or even fully preventing the inscription of problematic biases into the technology. Of course, not all future effects and unintended uses are foreseeable (especially since real life is messy, see the point made by Keymolen and Taylor in this book), and not everything can be prevented. However, a good start is to consciously implement certain values from the first stages of the design and to try to become aware of the values that we are unconsciously building into the technology. With this, "[t]echnological innovation can become *responsible innovation*"[emphasis original] (van den Hoven et al., 2015, p. 3). This places an active responsibility on engineers. Consciously focusing on the values inscribed into the design can help to ensure that the technology meets societal needs and it helps to reduce the risk of unwanted, unintended, or harmful effects. This also beneficial for the designers and engineers, because it may avert damage to their reputation when people consider the technology to be untrustworthy or harmful. Moreover, in some cases, designing technology in such a way that it promotes particular values is even required by law. An important law in this regard is the GDPR

(see the chapter by Gellert in this book for more information about this regulation), which requires agents who process personal data to engage in *privacy by design* (Art. 25, GDPR, I will return to this later).

21.3.2 Value-Sensitive Design

One of the ways in which we can consciously aim to deal with the problematic, as well as beneficial, non-neutrality of technology is by engaging in a manner of designing that is *value sensitive*. Several approaches have been developed for explicitly taking human values into account when designing technology. These approaches "share at least four key claims: values can be expressed and embedded in technology, technologies have real and sometimes non-obvious impacts on those who are directly and indirectly affected, explicit thinking about the values that are imparted in technical design is morally significant, and value considerations should be surfaced early in the technical design process" (Friedman et al., 2017, p. 65). The most well known of these approaches is *value-sensitive design* (VSD) (for an extensive overview, see Friedman & Hendry, 2019).

The general idea of VSD was developed around the mid-1990s (Friedman et al., 2017, p. 64). VSD is an approach to technology design that takes human values into account during the whole of the design process (Friedman et al., 2008, p. 76). Van den Hoven describes it as "a proactive integration of ethics—the frontloading of ethics—in design, architecture, requirements, specifications, standards, protocols, incentive structures, and institutional arrangements" (Van den Hoven, 2008, p. 63). VSD is ongoing under development and may always be (which does not have to be a bad thing). Its general methodology still faces some challenges (see, e.g., Friedman et al., 2017; Winkler & Spiekermann, 2018)—a few of these will be discussed below. Despite the challenges, overall, VSD is a relatively practical approach concerning value-conscious technology design and can be of significant value to those who are at the heart of the design process. VSD's methodology draws on inter alia the social sciences and human and computer interaction research (Friedman et al., 2017, p. 64). Its methodology **mixes empirical, technical, and conceptual studies** and applies these in an iterative and integrative manner throughout the design process (Friedman et al., 2008, p. 93). With this methodological mix, VSD takes an interactional stance and starts from the premise "that human beings acting as individuals, organizations, or societies shape the tools and technologies they design and implement; in turn, those tools and technologies shape human experience and society" (Friedman et al., 2017, p. 68).

► Example 2

Imagine that you want to develop software that helps people to spend less time looking at their smartphone. By means of **empirical analysis**, you can examine the context and experience of people's current smartphone use and get an idea of their wishes and problems. In order to examine this, you would ideally use quantitative and/or quali-

tative research methods from the social sciences, like interviews, surveys, and statistical analysis. Additionally, you can use such empirical analysis to test your own design. However, these analyses are not all that is relevant. Users may not always know what they want (especially beforehand), or be aware of all the implications of what they are doing and using, nor may you have sufficient data to oversee the bigger picture. Doing a **conceptual analysis** is therefore important to get a full(er) picture of the concepts and issues involved, like the values that play a role or the broader individual and societal implications of the technology. For this **conceptual analysis**, you draw on theoretic and philosophic research that sees to the main concepts and issues that relate in one way or the other to the (to be designed) technology. Let us say that for this software, you will be reading up on philosophical accounts of agency, autonomy, nudging, manipulation, and privacy. This can in turn inform your further empirical inquiries and your technological design. It is thus important to also perform a **technological analysis** of the technology. A better understanding of the technology can be achieved by analysing its concrete mechanisms and results, as well as by looking at already existing technologies that share certain similarities and assessing their impact. Your findings of the technical mechanisms can further inform and specify your conceptual and empirical analysis, which in turn can help you to improve the design. And so goes the process back and forth until you end up with a design that is well rounded and backed by research. ◄

21.3.3 Values

In the context of VSD, the term "value" refers to "*what is important to people in their lives, with a focus on ethics and morality*"[emphasis original] (Friedman & Hendry, 2019, p. 24). The focus thus lies on social and moral values, and not on economic value. In this context, we can think of values like human welfare, trust, privacy, fairness, autonomy, universal usability, safety, health, and environmental sustainability. The values potentially covered by VSD range from those that can be found in the diverse moral philosophical theories like deontology, consequentialism, and virtue ethics (see the chapter by Keymolen and Taylor in this book), as well as personal values like preferences of taste and colour, and conventions like protocol standards (Friedman et al., 2008, p. 94).

VSD tends to base its value selection and assessment on the experiences and opinions of the stakeholders. A key element of VSD is therefore to identify the direct and indirect stakeholders and their corresponding values (Friedman et al., 2017, p. 69). Direct stakeholders are people, groups, or organisations who directly interact with the technology in question (Friedman et al., 2017, p. 76). The indirect stakeholders consist of people, groups, or organisations who are affected by the technology, but do not directly interact with it (Friedman et al., 2017, p. 76). An example of a method to get a sense of the values at stake of the direct and indirect stakeholders is to conduct semi-structured interviews (Friedman et al., 2008, pp. 100–101). However, identifying the stakeholders can be difficult, and a failure to identify a particular group of stakeholders can lead to their exclusion as well as to the exclusion of particular values (Manders-Huits, 2011; Winkler & Spiekermann, 2018). Moreover, stakeholders themselves may not always be able to oversee the

impact of particular technologies and be able to recognise which of their values may be at stake in a certain context.

In some cases, it turns out that two or more conflicting values are involved. These conflicting values do not necessarily have to originate from different stakeholders: the same stakeholder can have multiple values that may pull the design in different directions. If there is a tension between values, it is important to take this into account in the design process (Friedman et al., 2017, p. 69).

▶ Example 3

An example of a tension between values is the likely tension between privacy and national or public safety: while privacy generally benefits from collecting and revealing less personal data, safety generally benefits from having access to more personal data. This tension played a pivotal role in the introduction of body scanners in airports. The goal of the body scanner is to increase safety by visually showing airport security staff where on the body people carry objects. For this, they scan the surface of the body, and this can display a rather accurate view of what the naked body of the scanned person looks like. This deeply infringes the bodily privacy of those scanned. Many of the body scanner developers took this privacy infringement for granted as a plausible sacrifice in the name of safety (Spiekermann, 2015, p. 169). However, it turned out that neither the general public nor the security staff and airport operators (who had to deal with customer complaints and feared a drop in customers) were all too happy with this privacy infringement (Spiekermann, 2015, p. 169). One company took both values—that of safety and privacy—seriously and sought to design scanner software that reduced the privacy infringement while maintaining its safety goal. In the resulting design, the display of the body was replaced by an abstract stick figure outline of a body with areas in which an object was located on the body marked. With this design, the company reached its safety goal while at the same time building in privacy safeguards in the software. By taking both values seriously and embedding them in the design, the company managed to capture the majority of the market (Spiekermann, 2015, p. 169). ◄

21.3.4 Legal Values and Design

A good source to find values that value-sensitive software design should ideally take into account, is law. In the context of software design, the GDPR is of particular relevance due to its focus on data processing. The GDPR provides us with a set of values that need to be taken into account on behalf of (the protection of) "data subjects" (see the chapter by Gellert in this book for a full explanation of the term "data subject") and society at large. Below are some of the main (derivative) values listed that can be found in the GDPR:

- Autonomy (see, e.g., informed consent, Recital 32, Art. 4(11), Art. 7)
- Privacy (see, e.g., control over own data, Recitals 7 and 68, Art. 17, Art. 21)
- Protection against power imbalance (see, e.g., Recital 43, automated individual decision-making, including profiling, Art. 22; purpose limitation, Art. 5(1)(b); data minimisation, Art. 5(1)(c); storage limitation, Art. 5(1)(e))
- Human dignity (see, e.g., Recital 4)

- Fairness (see, e.g., Recitals 39 and 60, 71, Art. 5(1)(a))
- Safety/protection (see, e.g., Recitals 1, 2, 51, 54, 78, and 108, Art. 1, Art. 6(d), Art. 6(f), Art. 25)
- Security (see, e.g., Recitals 2, 16, 19, and 49, Art. 5(f), Art. 32)
- Respect for the rights and freedom of individuals (see, e.g., Recitals 2 and 4)
- Human welfare (see, e.g., Recital 4)
- Transparency (see, e.g., Recitals 39, 60, and 71, Art. 5(1)(a))
- Economic prosperity (see, e.g., right to run a business, Recital 4)

What is interesting about the GDPR in light of this chapter, is that the GDPR even explicitly requires the embedding of some of its underpinning values in software design. Art. 25(1) of the GDPR on "Data Protection by Design and Default" states:

> Taking into account the state of the art, the cost of implementation and the nature, scope, context and purposes of processing as well as the risks of varying likelihood and severity for rights and freedoms of natural persons posed by the processing, the controller shall (…) implement appropriate technical and organisational measures, such as pseudonymisation, which are designed to implement data-protection principles, such as data minimisation, in an effective manner and to integrate the necessary safeguards into the processing in order to meet the requirements of this Regulation and protect the rights of data subjects (Art. 25(1), GDPR).

The data protection principles (the chapter by Gellert provides an extensive analysis of these principles; here, I will briefly touch upon them for clarity purposes) are listed in Art. 5 of the GDPR and state that personal data needs to be "processed lawfully, fairly and in a transparent manner in relation to the data subject ('lawfulness, fairness and transparency')" (Art. 5(1)(a), GDPR); the personal data can only be collected and processed "for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purpose (...) ('purpose limitation')" (Art. 5(1)(b), GDPR); the personal data needs to be "adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed ('data minimisation')" (Art. 5(1)(c), GDPR); the personal data needs to be "accurate and, where necessary, kept up to date; every reasonable step must be taken to ensure that personal data that are inaccurate, having regard to the purposes for which they are processed, are erased or rectified without delay ('accuracy')" (Art. 5(1)(d), GDPR); the personal data needs to be "kept in a form which permits identification of data subjects for no longer than is necessary for the purposes for which the personal data are processed (…) ('storage limitation')" (Art. 5(1)(e), GDPR); the personal data needs to be "processed in a manner that ensures appropriate security of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures ('integrity and confidentiality')" (Art. 5(1)(f), GDPR); and last, the ones controlling the data are held responsible and need to be able to demonstrate that they comply with the data protection principles ('accountability') (Art. 5(2), GDPR).

A significant role of these principles is to curb the personal data that can be collected and retained about a specific data subject. Here, the adage "knowledge is

power" comes to mind. In this context, the purpose limitation principle, the data minimisation principle, and the storage limitation principle are important restrictions that curtail the power imbalance that may rise between citizens and institutions and/or corporations that can aggregate massive amounts of information about them (see, e.g., Brouwer et al., 2011). On the other side of the coin, we can find measures in the GDPR that aim to better balance the playing field by ensuring that the data subjects themselves have sufficient knowledge about how their data is processed. For example, Art. 13(2)(f) of the GDPR seeks to ensure fair and transparent processing in the case of automated decision-making, including profiling, by requiring the data controller to provide the data subject "meaningful information about the logic involved, as well as the significance and the envisaged consequences of such processing for the data subject". Art. 22 of the GDPR sees specifically to automated decision-making and requires human intervention in cases where the produced decision can make a significant impact on the life of an individual. An example of a significant impact is the automated refusal of an online credit application without human intervention (Recital 71). The automated decision-making "should be subject to suitable safeguards, which should include specific information to the data subject and the right to obtain human intervention, to express his or her point of view, to obtain an explanation of the decision reached after such assessment and to challenge the decision" (Recital 71).[3]

The challenge of designing software value sensitively and trying to account for values like those referred to above is how to concretely embed these values in the designed product. For this, there is no one-size-fits-all method, as it depends on the values sought after and the software in question. Moreover, the "how" in itself is a topic of ongoing investigation. With regard to designing software in a privacy-enhancing manner, the article *A Critical Analysis of Privacy Design Strategies* by Colesky et al. (2016) can provide a valuable source of inspiration. The researchers identify several "privacy design strategies" that can be used to embed privacy by design. One of the suggested tactics is data minimisation; this links to Art. 5(1)(c) of the GDPR (see also the chapter by Gellert). It entails a selection in which the data that is not needed is excluded, stripped, or destroyed. Other strategies are access restriction and separation of data (Colesky et al., 2016). By isolating data collections, or by distributing them over different locations, the risk is reduced that the data is combined and provides a more detailed view on a specific individual and/or de-anonymises the individual. Another tactic they mention is abstraction (Colesky et al., 2016). If data is summarised or grouped on a more general level, the focus of the data shifts from particular individuals to a more generic level. Studies like the one performed by Colesky et al. can be an inspiration source for designers to come up with designs that realise their striven-for values.

3 However, requiring a human agent to be in the decision-making loop is no guarantee that the sought-after values are protected (Binns, 2019). Human intervention also has its up- and downsides: humans can discriminate intentionally and unintentionally. Additionally, if a human is added into the decision-making loop, there is the risk that the human merely "rubber stamps" the made decisions to validate their outcome and bypass the further requirements of Art. 22 of the GDPR (Veale & Edwards, 2018, p. 400).

Conclusion

Technology is not neutral. It can influence and shape people's perception, what they know, what they can do, the way they engage with the world, and the way in which society, governments, companies, and others engage with them. The innovative use of technology can therefore be highly valuable, but also highly problematic. Technology has therefore been the inspiration for both utopias and dystopias alike.

Value-sensitive design is an approach that aims to deal with the non-neutrality of technology in a beneficial manner. It focuses on actively aiming to incorporate certain values in the design of technology. While VSD is not without problems and challenges, it is a promising start for designing technology that aims to be on the utopian, rather than on the dystopian, side of things. In some cases, law even requires the embedding of certain values in the software design: art. 25 of the GDPR calls for the implementation of "privacy by design" and "privacy by default".

Designing value sensitively is not an easy task, especially because it is often difficult, if not impossible, to fully foresee the impact and use of a new technology. However, this should not stop us from trying. There is a pivotal role here for designers. Being aware of the views and assumptions that are necessarily built into the system design, they can try to do this in a conscious and value sensitive manner.

In order to start designing value sensitively, it can be of help to keep the following rules of thumb in mind:

1. Be aware of the inherent non-neutrality of what you are designing: think about what the technology adds to, takes away from, or changes in the current situation.
2. Identify which values you want to endorse with your design (e.g. you may want to design software in order to promote efficiency while at the same time safeguarding privacy).
3. Assess the impact of the design: does the technology benefit or negatively impact a specific group of people? And who are these people and what are the consequences for them?
4. Trace if you may unnecessarily or undesirably inscribe certain prejudices in the design (e.g. is the user you have in mind representative for the whole user group, or are you unconsciously designing the software in a way that only benefits a particular subset of users?).
5. Try to see if you can adjust the design in order to get rid of unintended bias or negative impact while promoting the values you want to endorse (i.e. test, evaluate, adjust).

How to (best) realise value-sensitive design (and in particular privacy by design and default given that these are required by law) is still—and with new technologies will always be—a topic of exploration and experiment. However, the first step is an awareness of a technology's non-neutrality, and a willingness to think about which values ideally should be protected in its design and how to achieve this. Hopefully, this chapter helps to readers with taking this first step.

Question
How can designers influence the non-neutrality of software?

Answer
Designers intentionally influence the non-neutral directionality of the technology by designing software to perform specific tasks and help users reach certain goals. They determine what a user can and cannot do with the program and thereby influence the non-neutrality on the user action level. This is set "in stone" in the code of the program. Additionally, the designers determine what a user perceives (in combination with the properties of the used hardware) when engaging with the application, thereby steering a user's experience and interpretation of the technology.

An important role here is played by the image of the user that a designer has in mind. An example is the design of a website with little text and a lot of images. Users with impaired vision that depend for their web surfing on an application that reads text out loud, will have a hard time navigating the website.

Additionally, designers can influence the non-neutrality of software unconsciously by embedding their own assumptions into the design. An example is an online credit card request form that requires one to make a photo of an identity card with a smartphone directly, and not allow the uploading of files. This assumes that all internet users have a smartphone.

Furthermore, by means of user manuals and marketing, particular views on and uses of a technology can be influenced and promoted, for example by suggesting that using a particular software application can improve health, social status, friendship, and efficiency, can reduce human errors, etc.

Question
What are the advantages of designing value sensitively?

Answer
First and foremost, engaging in value-sensitive design will help designers to innovate in an ethically responsible manner by front-loading ethical values in the design of the software: in the interface, architecture, standards, specifications, incentive structure, institutional embedding, default settings, user requirements, etc.

Furthermore, taking into account the interest of the various direct and indirect stakeholders and trying as good as possible to account for their values in the design will help to generate more societal support for the use of the technology. More happy people generally means more users and user engagement.

Also, because a thorough reflection on the potential impact of the software is a necessary part of designing value sensitively, the designers (or commissioning company) are less likely to be surprised by unforeseen consequences of the technology.

Question
Reread the text of Art. 25(1) of the GDPR cited in the text above. What is important for properly realising privacy by design?

Answer

21

Art. 25(1) of the GDPR does not call for a flat-out implementation of privacy by design. Instead, it calls for a delicate balancing of the available technical options, interests of those involved, the purposes of the data processing, and its context, risks, and impact on people. It is thus not only privacy that should be taken into account as a value in the design: other values, like safety, human welfare, and economic prosperity, should also be taken into account and balanced with privacy. Privacy by design thus means, simply put, an active prevention of any possible privacy infringement that is not strictly necessary for realising a particular goal. As the case of the body scanners discussed in ► Sect. 21.3 shows, a careful balancing can result in a design that is able to respect conflicting values to a considerable degree: while maintaining their goal of safety, the "stick puppet" body scanners also significantly reduce the privacy infringement on those scanned. With this balance, these body scanners are a good example of privacy by design.

References

Acquisti, A., Brandimarte, L., & Loewenstein, G. (2015). Privacy and human behavior in the age of information. *Science, 347*(6221), 509–514.

Binns, R. (2019). Human judgement in algorithmic loops; individual justice and automated decision-making. *Individual Justice and Automated Decision-Making* (September 11, 2019).

Brey, P., & Søraker, J. H. (2009). Philosophy of computing and information technology. In: *Philosophy of technology and engineering sciences*, Elsevier, pp. 1341–1407.

Brouwer, E., et al. (2011). *Legality and data protection law: The forgotten purpose of purpose limitation.*

Chabot, P. (2013). *The philosophy of Simondon: Between technology and individuation*. A&C Black.

Colesky, M., Hoepman, J. H., & Hillen, C. (2016). A critical analysis of privacy design strategies. In: *2016 IEEE Security and Privacy Workshops (SPW)*, IEEE, pp 33–40.

De Mul, J., & van den Berg, B. (2011). Remote control: Human autonomy in the age of computer-mediated agency. In: *Law, human agency and autonomic computing*, Routledge, pp. 62–79.

Feenberg, A. (2002). *Transforming technology: A critical theory revisited*. Oxford University Press.

Fogg, B. J. (1999). Persuasive technologies. *Communications of the ACM, 42*(5), 27–29.

Friedman, B., & Hendry, D. G. (2019). *Value sensitive design: Shaping technology with moral imagination*. Mit Press.

Friedman, B., Hendry, D. G., Borning, A., et al. (2017). A survey of value sensitive design methods. *Foundations and Trends® in Human–Computer Interaction, 11*(2), 63–125.

Friedman B, Kahn PH, Borning A (2008). Value sensitive design and information systems. In: *The handbook of information and computer ethics*, pp. 69–101.

Friedman, B., & Nissenbaum, H. (1996). Bias in computer systems. *ACM Transactions on Information Systems (TOIS), 14*(3), 330–347.

Gandy, O. H. (2010). Engaging rational discrimination: Exploring reasons for placing regulatory constraints on decision support systems. *Ethics and Information Technology, 12*(1), 29–42.

Giritli Nygren, K. (2009). The rhetoric of e-government management and the reality of e-government work: The Swedish action plan for e-government considered. *International Journal of Public Information Systems, 2*, 135–146.

Goldman, E. (2008). Search engine bias and the demise of search engine utopianism. In: *Web Search*, Springer, pp. 121–133.

Grimmelmann, J. (2010). Some skepticism about search neutrality. *The next digital decade: Essays on the future of the Internet*, p. 435.

Harjumaa, M., & Oinas-Kukkonen, H. (2007). Persuasion theories and it design. In: *International Conference on Persuasive Technology*, Springer, pp. 311–314.
Heidegger, M. (1954). *The question concerning technology. translated by William Lovitt in the question concerning technology and other essays.* 1977.
Hildebrandt, M. (2015). *Smart Technologies and the End (s) of Law: Novel Entanglements of Law and Technology*. Edward Elgar Publishing.
Ihde, D. (1983). *Existential technics*. SUNY Press.
Kiran, A. H., & Verbeek, P. P. (2010). Trusting our selves to technology. *Knowledge, Technology & Policy, 23*(3–4), 409–427.
Kitchin, R. (2017). Thinking critically about and researching algorithms. *Information, Communication & Society, 20*(1), 14–29.
Latour, B. (1993). *We have never been modern*. Harvard University Press.
Lazzarato, M., & Jordan, J. D. (2014). *Signs and machines: Capitalism and the production of subjectivity*. Semiotext (e) Los Angeles.
Lessig, L. (2006). *Code Version 2.0*. Basic Books.
Manders-Huits, N. (2011). What values in design? the challenge of incorporating moral values into design. *Science and Engineering Ethics, 17*(2), 271–287.
Mittelstadt, B. D., Allo, P., Taddeo, M., Wachter, S., & Floridi, L. (2016). The ethics of algorithms: Mapping the debate. *Big Data & Society, 3*(2), 2053951716679679.
Noorman, M. (2020). Computing and moral responsibility. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy, Spring 2020th Edition*. Metaphysics Research Lab, Stanford University.
Pariser, E. (2011). *The filter bubble: What the Internet is hiding from you*. Penguin UK.
Skitka, L. J., Mosier, K., & Burdick, M. D. (2000). Accountability and automation bias. *International Journal of Human-Computer Studies, 52*(4), 701–717.
Skitka, L. J., Mosier, K. L., & Burdick, M. (1999). Does automation bias decision-making? *International Journal of Human-Computer Studies, 51*(5), 991–1006.
Sparrow, B., Liu, J., & Wegner, D. M. (2011). Google effects on memory: Cognitive consequences of having information at our fingertips. *Science, 333*(6043), 776–778.
Sparrow, B. C., Chapman, P., & Gould, J. (2005). *Social cognition in the internet age: Same as it ever was?* pp. 273–292.
Spiekermann, S. (2015). *Ethical IT innovation: A value-based system design approach*. CRC Press.
Stiegler, B. (1998). *Technics and time: The fault of Epimetheus* (Vol. 1). Stanford University Press.
Stiegler, B. (2012). Relational ecology and the digital pharmakon. *Culture Machine, 13*.
Thaler, R. H., & Sunstein, C. R. (2009). *Nudge: Improving decisions about health, wealth, and happiness*. Penguin.
van den Berg, B., & Leenes, R. E. (2013). Abort, retry, fail: Scoping techno-regulation and other techno-effects. In: *Human law and computer law: Comparative perspectives*, Springer, pp. 67–87.
Van den Hoven, J. (2008). *Moral methodology and information technology. The handbook of information and computer ethics*, p. 49.
van den Hoven, J., Vermaas, P. E., & van de Poel, I. (2015). Design for values: An introduction. In: *Handbook of ethics, values, and technological design: Sources, theory, values and application domains* pp. 1–7.
Van Dijck, J. (2013). *The culture of connectivity: A critical history of social media*. Oxford University Press.
Veale, M., & Edwards, L. (2018). Clarity, surprises, and further questions in the article 29 working party draft guidance on automated decision-making and profiling. *Computer Law & Security Review, 34*(2), 398–404.
Verbeek, P. P. (2005). *What things do: Philosophical reflections on technology, agency, and design*. Penn State Press.
Verbeek, P. P. (2011). *Moralizing technology: Understanding and designing the morality of things*. University of Chicago Press.
Wihlborg, E., Larsson, H., & Hedström, K. (2016). "The computer says no!"—A case study on automated decision-making in public authorities. In: *2016 49th Hawaii International Conference on System Sciences (HICSS), IEEE*, pp. 2903–2912.

Winkler, T., & Spiekermann, S. (2018). *Twenty years of value sensitive design: A review of methodological practices in VSD projects*. Ethics and Information Technology. pp. 1–5.
Wittkower, D. (2014). Facebook and dramauthentic identity: A post-goffmanian theory of identity performance on SNS. *First Monday, 19*(4).