Humanities & Social Sciences Communications

ARTICLE

https://doi.org/10.1057/s41599-022-01206-4 OPEN

Empowered and embedded: ethics and agile processes

Niina Zuber[1], Jan Gogoll[1]✉, Severin Kacianka[2], Alexander Pretschner[2] & Julian Nida-Rümelin[3]

This article focuses on the structural aspects of the development of ethical software, and argues that ethical considerations need to be embedded into the (agile) software development process. In fact, it is claimed that agile processes of software development lend themselves specifically well for this endeavor. First, it is contended that ethical evaluations need to go beyond the use of software products and include an evaluation of the software itself. This implies that software engineers influence peoples' lives through the features of their designed products. Embedded values should thus also be approached by software engineers themselves. Therefore, the emphasis is put on the possibility to implement ethical deliberations in already existing and well-established agile software development processes. The proposed approach relies on software engineers making their own judgments throughout the entire development process to ensure that technical features and ethical evaluation can be addressed adequately to transport and foster desirable values and norms. It is argued that agile software development processes may help the implementation of ethical deliberation for five reasons: (1) agile methods are widely spread, (2) their emphasis on flat hierarchies promotes independent thinking and autonomy, (3) their reliance on existing team structures serve as an incubator for deliberation, (4) agile development enhances object-focused techno-ethical realism, and, finally, (5) agile structures provide a salient endpoint to deliberation.

[1] Bavarian Institute for Digital Transformation, Munich, Germany. [2] Technical University of Munich, Munich, Germany. [3] Ludwig Maximilian University of Munich, Munich, Germany. ✉email: jan.gogoll@bidt.digital

Software artifacts play an ever more important role in our lives. The constant presence of social media, decision-supporting technologies and information flows has blurred the boundary between digital products and the analog world. This development, in turn, prompts companies, users and regulatory authorities to focus increasingly on newly emerging ethical issues within said development. Software can influence user behavior through intentionally or unintentionally designed features of the operating function (Özdemir, 2020; Marthur et al., 2019, 2021). Any underlying features and their mechanics that are opaque or elude immediate perception exacerbate the paternalism debate (King and McCrudden, 2017). This issue is mainly discussed in connection with persuasive technology, ambient intelligence or ubiquitous computing (Reitberger et al., 2009; Ijsselsteijn et al., 2006; Ramos et al., 2008). The main points of this debate, however, must not be confused with acceptance or user-friendliness, which are already included in the usability approach. Ethical deliberation always goes beyond user-experience in that it always asks about the good life in general. These ethical questions can be captured by a structural analysis, namely highlighting established cooperative practices, for example cooperation between doctors and patients, businessmen and buyers, statesmen and citizens. These practices and their analysis foster an understanding of mutual expectations that are constitutive to a good life. In this way, then, can digital products be understood more comprehensively as socio-technical systems while their purely technological dimension is still adequately taken into account.

Rather, the technology itself must also be considered, meaning that any comprehensive analysis must identify the technological values (techno-generic values) embedded in a software and subject those to a normative analysis. In data analysis, for example, this includes questions regarding the handling of the data, the quality of the data set and its interpretation. Inaccuracies in these areas raise concerns of discrimination and privacy issues. Hence, software developers need to consider ethically relevant implications of their product design which, in turn, depend on the values often implicitly embedded into the software. It is important, however, to distinguish between implicitly established value attitudes and negative externalities that affect the use of technology in our lifeworld. This distinction supports software developers to classify the areas where they have to make decisions and assume responsibilities more easily.

Explicating and evaluating the values and attitudes that guide, and should guide, software developers in their work requires ethical deliberation. This is, essentially, a normative endeavor. Ethical deliberations comprise different ways of thinking such as descriptive approaches to identify values, evaluative modes to figure out qualities for individual cases and decisions to stop further enquiries. The need to integrate normative considerations in software development is also emphasized in the value sensitive design approach: The assumption that digital artifacts transport normative claims (Friedman and Hendry, 2019). However, it is not entirely clear how, when and who is in charge of these ethical reflections and ethically relevant design decisions within the overall process of software development. Consequently, ethical deliberations within this process need to be systematized and coordinated (Zuber et al., 2020), and engineers and companies must be supported and enabled to implement ethical engineering practices.

It is worthwhile to note that ethical engineering or ethical design is not meant to include machines that act autonomously: Machines are no moral agents (Nida-Rümelin and Weidenfeld, 2018). Machines conserve and perpetuate values or even transform and establish real life normative expectations. Thus, normative deliberation includes a disclosive moment (Brey, 2000a), an evaluative moment and a concluding moment. Normative deliberations comprise not only a juxtaposition of values, but also reveal normative constituents that compose everyday mutual expectations, i.e., how to adequately respond to certain requests made by others (Nida-Rümelin, 2019). Practice theory may help to better understand and include normative constituents while designing and developing a digital artifact: Our commonly shared lifestyles will be affected by the deployment of information technology. It is therefore worthwhile to ponder upon which of the normative stakes will be eroded, transformed, or substituted. This structuralistic aspiration is put into a system-theoretic point of view by Umbrello and Gambelin (2022): On the level of system requirements, the developer, in terms of their work attitude, and the user thus come together.

Identifying and understanding these normative facets require merging different modes of thinking each of which is oriented towards a different quality criterion: from technical functionality, economic cost–benefit considerations, to esthetic values and, of course, moral requirements. All these lines of reasoning and their respective focus must, if we want to prove them to be reasonable, meet logical requirements, such as freedom from contradictions and coherence which means to consider the relational conditionality of reasoning (Nida-Rümelin, 2020). Ultimately, these questions and requirements form what can be called a metatheory of software development: What kind of work-world conditions must be in place if we want to acknowledge the fact that digital artifacts have broader effects on our lives than achieving certain purposes in a technically or economically rational manner? In this paper, we argue that the comprehensive ethical deliberations, that are part of the required work-world conditions, need to be integrated into management processes if they are to become habitual development culture. Moreover, we will argue that agile processes are particularly suitable due to their work organizational structure and functional teams.

The aim of this paper is to show that agile processes have certain features and assumptions that may help to foster the efficient integration of ethical deliberation. We seek to cultivate a professional ethos that recognizes good normative design as an essential component of good development work. Therefore and because Principlism won't do, it needs to be made clear why to engage in normative deliberation as well as how to integrate it (Mittelstadt, 2019). This means they require the possibility of exercised judgment (1). First, we briefly elaborate on what we mean by ethical deliberation with regard to IT technology (1.1). Techno-ethical judgment requires autonomous persons, i.e., persons who perceive themselves as independent and whose independence is promoted within work culture. The empowerment of all participants who are involved in the process is thus a prerequisite for the successful integration of techno-ethical deliberation (1.2). Second, we discuss how the nature of digital artifacts and their societal implementation shape the technical mindset. The focus on the technological artifact is what provides relevant input that guides normative reasoning (1.2). We emphasize that these techno-ethical deliberations must be integrated into the work culture if they are to foster a professional ethic and a developer mentality that is not exclusively focused on technical functionality (1.3). Agile processes are suitable, we claim, because they empower engineers in their independence and thus leave room for ethical deliberations. Finally, we give five reasons why agile software development processes may help the implementation of ethical deliberation (2): (1) agile methods are already prevalent in software development (which eases the introduction of ethical deliberation due to low transaction costs), (2) their emphasis on flat hierarchies promotes independent thinking (an essential prerequisite to deliberation), (3) their reliance on existing team structures serves as an incubator for deliberation, (4) agile development enhances object-focused

techno-ethical realism, and, finally, (5) agile structures provide a salient endpoint to deliberation.

Empower the engineer: autonomy as a requirement for good normative design

Agile development processes naturally lend themselves to ethical considerations because they empower developers to work in small teams and solve problems independently, without tight central control (cf. Spreitzer, 2008; Tariq et al., 2016). This approach gives developers substantial leeway in influencing the design of the system. However, agile software development is no silver bullet. Agile methods do, for instance, have drawbacks compared to traditional software development methods (see for example, Cho, 2008 or Savolainen et al., 2010), especially when developing sizable and complex systems. There are many situations in which agile methods are difficult to implement. Examples include highly regulated industries including medical, aerospace, and automotive, where regulation often demands specific processes and elaborate documentation; the design of physical products where specifications often need to be decided on a long time before software development begins; large-scale projects; or corporate cultures that are highly hierarchical (e.g., Poth et al., 2020). One can point to four reasons that might explain these drawbacks: First, agile methods prefer code over documentation which is a problem in systems that legally require extensive documentation in order to satisfy safety and compliance demands such as medical software or automotive systems (Turk et al., 2002). Second, while agile methods work well for small teams, scaling them to systems where multiple teams are interdependent on one another has proven to be difficult. Third, the agile processes themselves are sometimes seen as too much of a burden to developers—demanding an excessive amount of meetings and other interactions that negatively impact a developer's ability to focus and these obligations bind resources that could be used for other purposes. Fourth, as alluded to in the adage that "agile development means replacing one bad developer with two good developers", agile processes require capable and motivated employees (Solinski and Petersen, 2016). Of course, capable and motivated individuals usually perform better than their less capable and motivated colleagues in any situation. For agile methods, thus, it has to be shown that they have the ability to further support developers because of the nature of this development style. We will provide reasons for this claim regarding the implementation of ethical deliberation in the section "Implementing ethics in the agile softwaredevelopment environment".

Agile development practices work particularly well in contexts where a small team of developers builds a product and requirements for said product are subject to frequent change. Here, the iterative nature of agile processes and the close communication with the customer improves the end result, because customers get to see early interim results and can thus help clarify their requirements. To oversimplify, true agile practices are more often found in teams that develop websites or phone apps than companies that develop braking systems or EEG monitors. Traditional software development often incorporates a reduced version of ethical deliberation in the form of a technology assessment or dedicated safety or security analysis (Schneier, 1999; Ruijters and Stoelinga, 2015; Grunwald, 2018). While many technology assessments concern risk forecasts, it is only since the early 2000s that ethical technology assessments came into play such as the VDI 3780 standard (Verein Deutscher Ingenieure, 2000) or Palm and Hansson's eTA approach (Palm and Hansson, 2006). In regard to the ethical design of software systems the IEEE 7000/D3 (2020) focuses on team members' competencies that are necessary in order to consider and deal with ethical issues while developing software. In doing so, they stress that ethical theories help in evaluating and identifying ethical values (IEEE 7000/D3, 2020, pp. 13–22). This is why, McLennan et al. (2020) stress to include an ethicist in development teams to foster ethical considerations during development. They argue that "the gold standard for embedded ethics integration would be an ethicist, or a team of ethicists, as a dedicated member of the development team" who—rather obviously—"possess[es] competence in ethical inquiry" but also "ha[s] domain-specific understanding and knowledge of the area of technology development in which they will be embedded" (McLennan et al., 2020). When it is not possible to include an ethicist directly into the team due to financial reasons or lack of available (full-time) ethicists, they propose an indirect approach with ethicists working—for instance—at universities to support ethical inquiry for the project. Larger tech companies have already begun to integrate ethicists into their workforce (Metcalf et al., 2019). Whether or not the supply of such experts is enough to staff every development team or at least provide part-time assistance remains to be seen. In any case, the focus on enriching the development process itself with ethical deliberation seems very promising as a formal process within the organization may also facilitate ethical awareness (Rottig et al., 2011). This has the additional advantage that all roles (developer, designer, manager, etc.) with their respective backgrounds are involved.

Normative deliberation. Values are often regarded as the key starting point to enable normative deliberation in software engineering (Spiekermann, 2015; Senges et al., 2017; Spiekermann and Winkler, 2020). This reasoning also inspires Codes of Conduct and Codes of Ethics as a means of establishing certain values that either shall not be violated by software or shall be taken into account in development. Codes of Conduct are certainly useful for establishing a set of (potential) values. For the development of ethical software, however, they do not constitute more but a mere starting point (Gogoll et al., 2021). Codes of Conduct typically comprise a set of higher-level regulative ideas, whose status is more or less self-evident, but they do not set forth a method of how to weigh these different ideas and to solve potential conflicts between them. They are to a large extent neutral regarding different theories of normative ethics, which is an advantage in terms of acceptability, but a disadvantage in terms of practicability. Hence, they cannot be more than the starting point of normative deliberations, especially when we consider that software is deployed in almost all contexts today. Not surprisingly, the characteristics of software also turn out to be very context-specific (Briand et al., 2017), as does the practice of software engineering. It therefore seems utterly unrealistic to expect a one-size-fits-all solution, i.e., a solution applicable to any kind of software, for embedding ethical values into software or applying ethical rules to the development process. The fact alone that software is context-specific requires a customized approach for each new software product.

Values are ideals to which people aspire. There are different types of values, economic values are motivated by economic reasons such as efficiency and profit, political values by political reasons and so forth. Moral values are motivated by moral reasons which means that their motivation goes beyond any personal interest an agent may have. Moral reasons often vary across cultures and can change over time. They can, however, be transformed into universal laws or incorporated into morally desirable behavior, i.e., virtues. To stress this point: A value such as *care* is displayed in a concrete principle such as "do not harm others". This can be formulated as a right once it has been recognized as a legitimate claim, such as the Offenses against the Persons Act. Alternatively, it can be expressed in a moral norm

which commands that you shall not abuse anybody, neither physically nor psychologically.

Good normative products display, foster or promote desirable values, norms, and laws., i.e. perpetuate desirable practices. The effects they have on our lifeworld goes beyond mere avoidance of negative externalities; meaning that normative deliberation in the context of these product's development goes beyond evaluating the consequences and trade-offs. Good normative design stresses the various modes of normative thinking that could and in fact should apply. Our approach of good normative design therefore systematizes normative deliberation and ties it back to management structures (Zuber et al., 2020).

Morally illegitimate IT entails software products that might, often accidentally, exclude people due to their capabilities; make people change their daily life without their consent; display information consent forms in such a way that the user feels overwhelmed; or nudge people to spend hours on their homepages or apps (Harris, 2016; Coeckelbergh, 2021; Mathur et al., 2019, 2021). Conversely, normative desirable software systems include aspects of integration, minimization of discrimination, reduction of sexism and racism, fostering mutual respect, inclusivity, trust, sustainability etc. (cf. Nissenbaum, 1999; Friedman et al., 2008; Zwart, 2014; Nida-Rümelin and Weidenfeld, 2018; Steed and Caliskan, 2021; van Wynsberghe, 2021). This quickly leads to an endless list of values that are considered relevant depending on the nature of the technology, *techno-generic values*, or the context of use, meaning *structural values*. So, it comes as no surprise that what makes a normative desirable products is notoriously difficult to identify and any ethical deliberations in that regard cannot be realized by using checklists or with the help of predefined answers (cf. Gogoll et al., 2021).

Instead, we need to identify and evaluate relevant ethical aspects. Relevant aspects in this sense may stretch across all phases suggested by value sensitive design and applied to agile as is done by Umbrello and Gambelin (2022). *Still, we must outline the stages of normative thinking that are necessary for each phase and map those onto the development process*. This requires concept of normativity that is responsive to the individual, intersubjective and collective characteristics of an artifact and how these characteristics are affected by how and where the artifact is used: Communication tools, for example, influence individual perception, i.e., what a person perceives to be fake news and how tolerant they are towards hate speech, intersubjective aspects may be particularly relevant in cases of cybermobbing; and of course, communication tools and any normative deliberation on them must consider their influence on the collective assessment of normative democratic features. In this case, this kind of normative deliberation merges hermeneutic phenomenology, structural analysis and practice theory. Social scientific methods such as stakeholder analysis and focus groups that all belong to the domain of acceptability research will further enrich the process of value elicitation. Let's call this the *epistemic phase of value elicitation*. Once accomplished, values need to be prioritized and evaluated which in turn needs *evaluative thinking*. In doing so, we exercise judgmental reasoning (cf. Rohbeck, 1993): we apply general principles (wisely) to particular cases. How we link principles and their application in particular circumstances cannot be achieved through deduction or subsumption alone. It is this rational-hermeneutic deliberation that cannot be understood in any algorithmic pattern. This kind of deliberation requires experience in practical principle application as well as the necessary working conditions (for the latter see also Umbrello and Gambelin (2022)). The *concluding phase* emphasizes habitualized willpower and decisiveness, which are to be grasped as personality traits rather than cognitive abilities. This approach is closely linked to bottom-up ethical approaches that refrain from Principlism in ethics (Dancy, 2004, 2018)—a dominant trend in modern times since the rise of European rationalism. We call all necessary normative deliberations the *exercise of techno-ethical judgment*. This kind of judgment needs to be part of a software development mentality in general and in this sense a daily routine: It combines factual technological knowledge with practical normative reasoning.

It seems reasonable to address values in IT within the framework of virtue ethics, i.e., to think about whether a particular information technology fosters or undermines e.g., moral, sustainable or autonomy-respecting practices. In virtue ethics moral practice is defined as a collective action whose purpose is shared and designated as valuable, e.g., honesty, friendship or care (cf. MacIntyre, 1981). If IT hinders people from establishing and nurturing caring relationships and even brings about behavior that makes it easier to stalk, bully, humiliate, or defame people, it is certainly worthwhile to rethink its design or—in the worst case—think about whether to deploy it at all. Virtue ethics highlights the fact that people who show the trait of being a caring person will act accordingly without constantly having to cognitively deliberate whether they want to be a caring person.

This makes virtue ethics a viable approach in the domain of software systems, in which desirable behaviors, in the classical sense *virtues*, are applied to techno contexts (cf. Vallor, 2016; Reijers and Coeckelbergh, 2020). Once desirable attitudes are formulated, they need to be promoted by practices. Vallor (2016) focuses on 12 techno-moral virtues: 1. honesty, 2. self-control, 3. humility, 4. justice, 5. courage, 6. empathy, 7. care, 8. civility, 9. flexibility, 10. perspective, 11. magnanimity, 12. techno-moral wisdom. In contrast to values applicable to a technical design, virtues pertain to the characters of the developer as well as to the user or other stakeholders. The habituation of those 12 good techno virtues will promote a good life and reduce at its best moral wrongdoing through the deployment and use of technological devices. Thus, Vallor is not exclusively addressing software developers, but the entire community. Therefore, desirable virtues need to be taken into account in the design of a product insofar as desirable practices should not be undermined by the development, deployment, and use of IT. In this context, virtue ethics may be an interesting move in information ethics, as it places the same demands on users and developers. Developers should develop those desirable virtues, strengthen and live them through work practices. The same is true for users: They should be enabled to live their lives in line with their good attitudes and shall not be corrupted. Virtue ethics can enrich systems theory approaches with the idea of the good life. This, in turn, means that management and development cultures should encourage and foster techno-virtuous behavior by introducing desirable practices into the development process. The ultimate goal is to focus on a development process that fosters normative deliberations, which in turn means to design products in such a way that they will not undermine morally good user behavior, but rather support desirable outcomes. Following this line of thought promotes the idea that any deliberation, decision-making and action within the context of software development as well as handling the product throughout the entire lifecycle must be in line with techno virtues reflection.

Thus, ethical issues need to also be addressed from the engineers' perspective. We need to foster practical reasoning with regard to software engineering. On the one hand, this is a question of curricula and education (cf. Shen et al., 2021; Anderson, 2017; Tavani, 2013, esp. Chap. 3). On the other hand, it is relevant for enabling or encouraging normative reasoning within the working environment (Umbrello and Gambelin, 2022). Ethical deliberation requires opportunities to explicate and navigate the context between values, goals and specific requirements that guide

implementation (for an overview of goals in requirements engineering, see van Lamsweerde (2001)). This skill should be learned and practiced, esp. because normative competence is not only an epistemic endeavor, which would mean that the main goal is an increase in knowledge, but rather a technique: It is exercising judgment, or in Vallor's terminology, disposing of *technomoral wisdom*. It must be emphasized that this power of judgment is also an attitude. This enables the individual to acquire an ethical awareness about the necessity for normative deliberation if the developed products are to be moral, sustainable, or promoting democratic values etc. This means, then, that practical reasoning is not a purely cognitive process, but also a *hexis:* a stance towards the world. We understand virtues (*arete*) rather in Aristotle's classic sense that a virtue (*arete*) is not only a result of habituation (*ethos*), but also entails a krisis (*decision*) and a hexis (*attitude*) (cf. Nida-Rümelin, 2020). In modern virtue ethics this threefold constitution of virtues is often left out and virtues are reduced to result from social habituation only (cf. McIntrye, 1981).

Normative deliberation requires autonomous thinkers who are aware of their ability to integrate different normative reasoning requirements into one judgment. This is why, any sense of being externally determined or limited in one's deliberation must be reduced as much as possible, so that people develop their own ideas. Developers need to be empowered and encouraged to not reduce their thinking and development skills to technical functionality only. In order for strong autonomous thinkers to be effective, they must develop their own ideas and must not be externally determined, neither by processes, roles nor hierarchies. These favorable working conditions ensure developers are bound to normative principles and to normative thinking because they operate along processes that are enriched with ethical deliberations. These conditions, ultimately, lead to the formation of a work ethic that understands normative deliberations as a daily work routine. Our approach, then, offers a process responsive to the heterogeneity prevalent in software development and the consequential lack of shared objectives in the profession. Both issues are discussed in Mittelstadt (2019), who identifies the lack of "common aims and fiduciary duties" as well as missing "methods to translate principles in practice" as difficulties to introduce ethical standards. Only development processes that are instilled with ethical deliberations shape a professional ethos that promotes good ethical behavior as a routine. Hence, rather than Principlism we strive for desirable attitudes (arete) in jointly supported work cultures (praxis).

Techno-ethical deliberation: How software steers normative deliberations. If the software engineer is expected to deliberate on normative issues and not only to focus on technical issues in terms of feasibility and functionality, digital artifacts should be understood as purporting or implying values itself. However, this is not to be misunderstood as the digital product creating its own moral values. Rather, the value attitudes of the respective designer are incorporated in the digital artifact. Such an understanding of technology is called the *embedded values approach* (Friedman and Nissenbaum, 1997; Brey, 2000b; van Wynsberghe and Moura, 2013). In John Moor's (2005) terminology: the subjects of this analysis are *ethical impact agents* and *ethical implicit agents*, that is, those software systems that either cause external negative moral effects or those in which (distorted) moral expressions have already been implemented. Accordingly, the analysis excludes attempts to develop *explicit and completely ethical agents*, i.e., artificial agents that are supposed to be able to autonomously perform deontic deliberations in a certain domain or across many situations, i.e., agents that can deliberate morally and justify their decision in a well-founded way.

Normative deliberations need to go beyond the value neutrality thesis of technology. Only in doing so, are we able to understand and think in a normative adequate manner about information technology and good normative system requirements. Therefore, it is crucial to debate the ontological particularity of digital objects not only in technical terminology.

Information technology comprises preset products over which the user has no complete control (cf. Hubig, 2015; Grunwald, 2015): Take washing machines, search engines or assistance systems in robotics. They already contain many assumptions about the water consumption, the power consumption or about the placement procedures of the search results up to the external design of a robot, which can be anthropomorphic or zoomorphic and thus simulate a lively encounter. All these features are not neutral insofar as they contain or convey values. The way in which the user can interact with this kind of digital technology is already partially predetermined by the technology itself which means that the user's autonomy is restricted in the interaction. B.J. Fogg (2002) refers to this characteristic of many IT-systems as *persuasiveness* insofar as the autonomy is affected by certain types of social clues that are triggered by the integration of technology within social interactions (e.g. social dynamics, social roles, etc.). This concept may be partially transferable to classical technology insofar as every technology requires certain behaviors, because certain reactions are usually necessary for technology to be functional. In the case of information technology, however, many pre-formed properties and infrastructures already have certain values which are not immediately obvious, and which cannot be influenced from the outside (or only to a limited extent).

Additionally, information technology is implemented throughout many domains of daily life and its determining force is often quite invisible, impenetrable, and multidimensional and thus difficult to grasp. Those phenomena are subsumed under the concept of *opacity* (Brey, 2000a, 2000b; van den Eede, 2011). It is claimed that information technology is of an opaquer nature than many traditional tools, such as doors, hair dryers, or hammers. Information technology is not restricted to one specific working context, precisely because it can serve a variety of purposes. Just think of the telephone receiver that one puts down so that one is not reachable at a given moment. Surely that is not the intended purpose of the telephone receiver. This surplus of purposes (Rohbeck, 1993) makes an ethically defined scenario difficult sometimes, since the use and orientation can change at any time (Vallor, 2016). Even the speed with which software products are developed makes them rather unique. Moreover, executing *mind work* (Agar, 2019), information technology substitutes or supports not only a pure physical force but is often placed within the very core of humanity as such: It strikes into the marrow of sociality, emotionality and reason. This can be illustrated by information technology supporting communication (e.g., social media or video calls) or decision-supportive analysis and recommendation software assisting medical diagnosis, application processes or parole conditions. Many information technologies transform the way we live precisely because they impact that which is genuinely human: how we express our emotions, how we judge, how we evaluate trust relationships, and much more. Thus, information technology can realign reciprocal normative commitments that sustain our sociality (cf. Nida-Rümelin, 2019). These are all very compelling reasons for demanding and imposing ethical deliberation, but also for why this endeavor is difficult and complex.

Hence, in the case of information technology, it remains unclear in many cases which questions are the right ones to ask and to subject to normative deliberation. This is precisely because emergent technologies are not yet embraced as practices and thus are not assignable to the stakeholders' outcomes (Vallor, 2016).

Practices are fixed modes of actions that aim at specific desirable values, such as to foster friendship or benevolence. Information technologies are still too dynamic and hence do not present options to act but rather different forms of life (Vallor, 2016; Nida-Rümelin, 1996, 2009). Vallor (2016) put it nicely when highlighting the fact that to ask whether "Is Twitter right or wrong?" (p. 27) or even whether "Tweeting is right or wrong?" (pp. 27–28) are meaningless, empty questions, since they cannot take into account any particular existing contexts of shared actions. To make this point clearer: The rightness of an action cannot be determined by one single criterion or principle, the action's consequences or the agent's motivation alone. The desirability of the decision needs also to be evaluated against existing valued structures and norms. An ethical deliberation has to grasp questions such as: Does Twitter promote the value of "caring", "courtesy", "friendship" if to take structuralist values seriously: For example, questions such as which values need to be taken into account and which practices need to be addressed within communication platforms that allow for short statements? How does Twitter respond to the demand of a respectful handling of each other despite diversity? Therefore, in the context of many ITs, it is not only necessary to ask about individual moral actions, but about the moral goodness of whole forms of life (Nida-Rümelin, 2005, 2009), i.e. to address them from an intersubjective and collective point of view.

Indeed, there are some important issues that need to be taken into consideration regarding the integration of ethical deliberations into the daily work environment. Firstly, who do we need to deliberate on ethical issues while developing and when should this deliberation be conducted? Since this is a procedural endeavor it needs to locate specific organizational moments in which the deliberation should be conducted, e.g. in an agile work environment, retrospectives or reviews might serve as an eligible place (Umbrello and Gambelin, 2022). Another aspect, which is also a procedural task, is to ask how such an ethical deliberation might itself look like? The latter is not a question of software engineering or requirement engineering as no requirements are formulated beyond technical functionality (Vakkuri et al., 2021). It is clear that good normative design must rest on both procedural aspects taken together. Methods like ECCOLA take the procedural aspect of implementing ethics into the development seriously by introducing a card deck that displays issues relevant to a specific ethic, such as AI or blockchain (Vakkuri). Within this approach the rationale and justification of normative deliberation, i.e. which values, norms or rules ought to be taken into account, is left aside, since "[f]or every iteration, the groups would select the ECCOLA cards they felt were the most relevant for the requirements of that iteration" (Vakkuri et al., 2021, p. 11). While Vakkuri et al. embed ethics into processes using familiar methods of software engineering it remains unclear what guides such an ethical deliberation besides sheer luck of looking at the right card at the right moment. Therefore, it is of great importance to combine such a software engineering approach with ethical frameworks that support the identification of relevant normative aspects. For various societal subfields or branches so called ethical frameworks exist ranging from medicine ethics (van Bruchem-Visser et al., 2020) to ethics of technology, e.g. ETICA (Stahl and Flick, 2011). Such frameworks rather focus on making an ethical evaluation transparent than on discussing its correct implementation. Besides academic approaches, business canvas exist that try to give practical guidance and try to tackle normative deliberation in a less complex manner, e.g. the Open Data Institute. This is not the place to discuss the findings, benefits or shortcomings of ethical frameworks. Still, the problem remains that only the cornerstones of deliberation can be articulated rather than an exact procedure regarding how we need to communicate and discuss values or how values are to be weighted. Therefore, the performance of *evaluative rationality* cannot be described due to the very nature of the matter: We cannot capture the procedure of normative reason, but only exercise it. This exercise, in turn, must be trained, which is why underdetermined principles, values and laws are nevertheless of great help as orientation and provide guidance. Taking a stance is strenuous, but it belongs to the core of moral autonomy.

Embedding ethics into software development. In the domain of software development, agile methods (Scrum, extreme Programming, Essence, etc.) offer many features that enable the development of ethically informed products throughout the entire development process. If ethical deliberation is to become a part of software development culture, it must be compatible with the processes used by software developers, designers, and operators; any other approach bears the risk of ethics to be ignored or reduced to a mere window-dressing effort. Ethical deliberation of information technology (IT) requires in-depth knowledge of the technical feasibility of normative criteria. However, even if the required expertise existed, it has, so far, not been explicitly integrated into an entrepreneurial culture, although some scholars have begun to tackle this question. Umbrello and Gambelin (2022), for instance, propose a systems-theoretical approach to the interactions and repercussions of individual domains. In this way, they emphasize that the developer must also see herself as part of the system. Likewise, the developer must focus on the various facets of the digital-technical system, i.e., cannot view the artifact exclusively as a technical object. Our approach is grounded in rationality theory to firstly, highlight the structures of thought to which different spheres of action belong, and, secondly, to emphasize their logical consistency and practical coherence (Nida-Rümelin, 2000, 2020). This enables an approach of descriptive-sociological structural analyses as well as discussing normative questions (Nida-Rümelin et al., 2021).

Umbrello and Gameblin (2022) show how techniques suggested by value sensitive design may neatly be integrated into agile processes using the method of "value flows and dams" as an example. We do agree with their overall argument. However, we want to justify why agile processes—on any level—are particularly suited for the integration of ethical deliberation due to their empowerment aspects. While this article does not explore any specific concepts, future research may systematize ethical deliberations by highlighting the relevant concepts we need to consider when designing software. Broadly speaking, the focus needs to be put on arguments and deliberations, not only on methods to gather more information. To this end, the difference between theory, method and techniques must be illuminated and their place in the agile process must be determined.

Ethical deliberations must be integrated into work processes in such a way that they are perceived as something positive. Individual software engineers need to be empowered and encouraged to reason normatively and be trained in ethical communication skills. We focus on software engineers because we address ethical issues that result from technical designs: Finding technical solutions for normative questions will not solve all ethical questions related to information technology. It will, however, promote more thoughtful designs and a responsible art of engineering in general. At the same time, it must be emphasized that those deliberations require structural embedding, but cannot be replaced by it, i.e., what is required is a process that enables and fosters ethical thinking. Therefore, we suggest including normative deliberation into development processes and thus make it part of documentation as well as

quality assurance. Ideally, this approach links normative deliberation to the whole software lifecycle, for example by embedding agile ethics into DevOps methods, which integrate the development and the operation of software into a single methodology (Ebert et al., 2016).

Such a proactive stance towards information technology is worked out e.g., by Brey (2012), Floridi (2008, 2011; Russo, 2012) and Reijers and Coeckelbergh (2020). They highlight, independent of their different theoretical foundation, the ability to influence real-life issues by changing technical design features. In this way, they expand the scope of action of the individual software engineer. A similar approach called Values in Design or Ethics by Design (Simon, 2016) has become popular since the 1990s. This ethical push emerged primarily in software engineering (van den Hoeven et al., 2015) to enable the deliberation of *embedded values*. The idea is that technology conveys values that have lifeworld implications, more precisely, that these values have moral consequences in the real world. Furthermore, these values implicitly embedded in technology reflect and perpetuate the attitudes and actions of its developers. Therefore, the focus throughout the entire development and deployment process is put on the project team's actions and decisions, which includes the software developer. Value sensitive design (Friedman et al., 2002), Ethics by design that primarily addresses AI issues (Richards and Dignum, 2019), values in design (e.g. Manders-Huits, 2009) or values in play (Flanagan et al., 2008; Flanagan and Nissenbaum, 2014) are similar methods to approach the embeddedness of values in digital technical products. They all oppose (strong) technological determinism and affirm individual autonomy to control technology and its outcomes. Hence, software engineering teams and companies have at least partial power and control over the design of their digital products, i.e., software engineering can intentionally exert influence on individuals (cf. van den Hoeven, 2015). It is important to stress that incorporated values do have some sort of moral impact on real-life structures and are therefore subject to normative evaluation.

Embedded values approaches are a very practical enterprise that is intended to support the software developer in translating normative attitudes into technical objects. It can therefore be understood as a form of hermeneutics. Since the approach originates from software engineering, it is less theoretical or meta-theoretical, but brings with it a plethora of individual cases, which in themselves do not provide any real normative guidance (e.g., discussing one particular issue such as privacy by design or accountability). Thus, those approaches remain rather a second-degree decision aid, even in the case of value sensitive design that presents the most theoretical fundament (Friedman and Hendry, 2019). As values and normative principles are identified and prioritized, which is a theoretical and empirical endeavor, methods that integrate values in design will become necessary and helpful (cf. Grunwald, 2015). However, little thought is given to the work environment or competences that are needed for such an approach to be successful (cf. Mittelstadt, 2019; Umbrello and Gambelin, 2022).

Therefore, introducing ethical deliberation into the software development process from the very beginning is strongly recommended when building ethically sensitive machines and software. Of course, institutional support for ethical behavior is not only a question for software producers or developers (Gogoll et al., 2021). Yet, in comparison to other professional ethics such as medical ethics or business ethics, software engineering ethics needs to address peculiar demands in dealing with normative aspects simply because actions and decisions by software engineers are instantly translated into technical requirements and thus are so often perpetuated millionfold. Likewise, ethics for software systems cannot be defined exclusively in terms of a social domain with its associated value systems, as is possible, for example, in business ethics or medical ethics. Due to the diverse use in the different areas, normative orientation standards must be reconsidered and adapted. Moreover, since software engineering lacks institutional sanctions and an overall professional commitment to consider public welfare (Abbas et al., 2019), it is essential that the development team itself is capable of knowing how to guide itself through normative concerns. Hence, ethical deliberation and technical skills are competences that are required to achieve good normative design.

Thus, a humanistic, anticipatory ethics of information technology with regard to a professional group (software engineers) is needed, which includes *techno-ethical judgment* and approves of a procedural character such as rational discursive moments (Ott, 2005). In order to understand such a humanistic approach as professional ethics, ethics must be understood as a guided practice that strengthens an *ethos*, i.e., habits that most professionals recognize as their purposes for action. For this, one needs a proactive ethics, i.e., the understanding that normative considerations must play a role from the very beginning and throughout the whole product lifecycle in order to be able to specifically develop normatively desirable artifacts. Since technology falls into the realm of possibility and uncertainty, and is thus intrinsically dynamic, we can only capture this through a processual ethics, i.e., rethinking the conditions, or to use Shilton's (2013) designation to introduce *value levers*, that make such a work ethic possible. This means that companies need to empower software engineering teams to undertake ethical deliberations, i.e., taking values, social goals and collective goods seriously in their considerations. In doing so, software engineers are bound to reflect in an ethical manner since ethical deliberations are part of the management process and thus may become a daily routine. This may overcome the problem of lack of commitment as identified by Mittelstadt (2019). Moreover, bringing normative deliberation into a systematic schema that is directly focused on software systems (Zuber et al., 2020) will reduce moral distress and help to overcome anxieties on the developers' side. This will help establish a professional ethical approach and institutionalize practices. Precisely because Principlism, in terms of setting forth certain desired value attitudes to which a digital product should be aligned, is not sufficient for practical development work (Mittelstadt, 2019). On one hand, there is a need for the exercise of techno-ethical judgment and, on the other, for procedural integration into management structures. The latter is now discussed.

Implementing ethics in the agile software development environment

Although Talcott Parsons made the concept of agility part of organizational theory as early as the 1950s, it had only a limited influence on the 2001 *Agile Manifesto* set up by Kent Beck and other experienced software developers (Förster and Wendler, 2012). In their manifesto they formulated four basic value slogans as the essence of agility (see http://agilemanifesto.org):

- Individuals and interactions over processes and tools,
- Working software over comprehensive documentation,
- Customer collaboration over contract negotiation,
- Responding to change over following a plan.

The Agile manifesto prioritizes individuals, working software, customer collaboration, and responding to change over a strict plan that needs to be followed, processes that need to be implemented, or documentation that needs to be filed. The goal of these principles is to provide simple responses to constantly

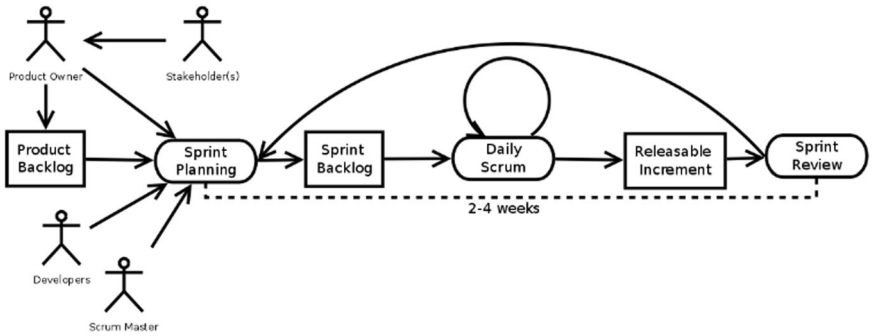

Fig. 1 A schematic view of Scrum.

changing external requirements and thereby avoiding costs generated by finalized but unsatisfactory software products—whether this is because of changing requirements, contexts, technological shortcomings or non-functional problems, which are often rooted in disregarding values. The objective formulated by agile methods is to produce working products while taking into account constantly changing environments. In agile processes, working (i.e., executable) products can continuously be presented to the customer who may test and evaluate the software immediately so that the current status quo of the product can serve as a basis for further deliberations (Flewelling, 2018). The idea of agility is rooted in the belief that a software system can hardly be specified up front and that it will always be necessary to react to unexpected changes, because of unforeseen environmental impacts, technological development or simply because the customer changed their mind. Thus, agility can be defined as provided by the Advanced Research Programs Agency (ARPA) and the Agility Forum as "the ability to thrive in an environment of continuous and often unanticipated change" (Sarkis, 2001, p. 88).

Due to its flexibility, agile processes have become the de facto standard for software development in non-regulated industries in recent years. According to a survey by digital.ai "95% of respondents report their organizations practice Agile development methods" (Digital.ai, 2020). The field of software engineering leads with about 37% of teams using the agile method to develop software. Arguably, respondents in these studies may overstate the level of agile adoption, e.g., how many teams are actually adhering to the complete toolkit of agile methods vs. only cherry-picking some aspects of it. Yet, it seems accurate that agility has "become a mainstream development methodology" (Shim and Lee, 2017) or even "the new normal"—especially when we take into account that many companies plan to adopt agile in the short and medium future (Koning and Koot, 2019; Hewlett Packard, 2017). Agile development methods such as Scrum, Extreme Programming or Kanban were used to anchor individual techniques such as user stories or pin boards in organizational cultures. By implementing them in work processes classic hierarchization (top-down approach) can be avoided in order to realize the values of agile programming. To analyze how ethical deliberations can be integrated in agile programming, we focus on one specific and also the most common (Digital.ai, 2020) agile process, Scrum, as depicted in Fig. 1. (cf. Schwaber and Beedle, 2002; Cao and Ramesh, 2008; Schwaber and Sutherland, 2011).

As outlined above, the core idea of Scrum is that customer requirements and development conditions will change. The best way to satisfy a customer is to deliver small incremental releases that will slowly converge with the customer's wishes in fixed time intervals, so-called sprints that usually last 2–4 weeks. (It is not by chance that this incremental approach also alleviates a fundamental problem of software engineering, namely the integration of different subsystems.) A Scrum consists of a product owner, who interfaces with the stakeholders (mainly the customer and the users. However, this could also include representatives of the government, NGOs or other groups with an interest or stake in the final product (Sverrisdottir et al., 2014)) and represents their interests; developers who implement the functionality; and a scrum master who is responsible to remove any work impediments and ensures that the development process runs smoothly. While in traditional development processes requirements are often prioritized at the beginning of the project, agile projects often change the priority of requirements or even remove them in their entirety. In Scrum, the product owner will be part of every sprint planning meeting and jointly with the developers decide which requirements will be part of the next iteration. Requirements that are not selected for the next sprint remain in the so-called product backlog, essentially a list of requirements, and will, if they have not become obsolete, implemented in one of the next sprints. It is important to note that the sprint planning meeting is time-boxed: for a 2-week sprint it should last 4 h and up to 8 h for a 4-week sprint. This means that the deliberations must come to an end and a decision must be reached. After this, if no major problems occur, the development team could focus on delivering code and not be stressed by making new decisions every day.

Of specific interest to our work is when and how requirements, which are normative in nature, are translated from being values or principles to concrete requirements to be implemented by developers. It is precisely here where the scope of normative deliberation can be widened, and, consequently, values and attitudes must be addressed and made explicit. However, the *Agile Manifesto* (2001) did not address ethical issues when introducing its agile principles. It does, for instance, highlight the need for the customer's or the product owner's positive attitude towards changing requirements and fluid prioritization in exchange with the software team; but this does not explicitly include public interests (Judy, 2009). This results in an incongruity in the formulation of technical, economic, and ethical requirements. Besides the integration of ethical deliberations in an agile work culture, it may also be important to foster moral attitudes such as whistle blowing or ethical peer pressure which are often neglected, after scandals such as Cambridge Analytica.

In the following, we will offer five reasons to illustrate why agile processes should be considered as a well-equipped platform to enable ethical deliberation during the development of software products.

Agile is already widely spread in the industry. Introducing any approach to implementing ethical deliberation to software development that would require practitioners to completely change their way of working would be an uphill battle. Fundamental changes to a management culture are costly (both in terms of money and time) and require strong commitments from

developers, designers and management (Gablas et al., 2018). Morality certainly is an important factor to consider in software development, but it is by far not the only one. Software engineers and designers want to build products; and companies have to create cash flow. Therefore, instead of creating an entirely new way of doing business, we suggest including ethics into a working style that is already widely used and has features and capabilities that promote and foster ethical deliberation.

In order to avoid transition costs and minimize costs of implementation in general, we should thus look for ethical deliberation to be implemented in an already existing management system. While agile offers more than just a high rate of adoption, prevalence is one of the necessary conditions when trying to put theoretical considerations into actionable practice. If a method of developing ethically guided technical products required a fundamental change in a company's workflow, it is hard to imagine that it would succeed. The agile method being widespread in software engineering makes a symbiotic approach auspicious and more likely to have a positive effect on development in the near future. Agility being anchored in the software industry eases the introduction of ethical deliberation because it is integrated into a well-known, learned, and already working method of development. Consider, for instance, the fact that agile already has a well-defined process of meetings and planning. It seems obvious that the addition of ethical deliberation and the documentation of it will decrease the friction as opposed to the introduction of new processes.

Furthermore, since the specificity of a potential ethical issue depends on the concrete product and situation which makes a one-size-fits-all approach to tackle ethical issues during the development impossible, the use of an already implemented development regime offers the advantage of providing risk-free leeway for testing out various mechanisms to further ethical deliberation within the agile framework. For instance, a company could experiment with the introduction of ethical deliberation within teams as a follow-up to the daily sprint (a regular short meeting) every fortnight or add user stories (requirements) that deal with the experience of minorities, different perspectives etc. Thus, firms would be able to learn more about when and to what extent ethical deliberation is necessary depending on a project's context, its domain as well as team size, seniority of developers, and other relevant factors (cf. Spiekermann and Winkler, 2020). This could all be done with comparatively little additional effort for the developers since it would be implemented into the already existing framework of agile.

It is important to note that for agile frameworks to work, strong support of management is a necessity (Dikert et al., 2016). The same is true for ethical deliberation within the process. If ethics is perceived as a part of agile software development, hopefully, it can also piggyback on management's support of agile in general.

Flat hierarchies offer autonomy to developers and designers. As we have argued above, ethical deliberation requires that software developers have certain degrees of freedom in their work. In an environment in which developers strictly follow orders, work on a narrowly defined problem, and are thus isolated from interactions with other steps or actors in the development process, there is little room for ethical deliberation other than on the level of management. However, the information asymmetry that management faces especially in the IT domain, might render many ethical issues which might become obvious during the development process inaccessible to the decision makers at management level so that they remain unsolved. If we want to introduce moral attitudes within the development process and include developers in the act of detecting ethical pitfalls, incorporating stakeholder values, and developing normative desirable software, we must provide developers with greater freedom in the development process. Agile as a process offers just that. In fact, empowerment and autonomous decision making are seen as a key factor in regard to forming truly agile teams (Kidd, 1994; van Oyen et al., 2001; Mudili, 2017).

If software developers and designers work in clearly defined hierarchies with a narrowly defined scope of action, there is little to no place for critical thinking. For example, employees who expect primarily monotonous work are more likely to accept organizational misconduct or attempt it themselves (cf. Staffelbach et al., 2014). In such contexts, critical thinking and the additional motivation required to attempt this kind of thinking are perceived as external factors and therefore something that can be delegated. Conversely, intrinsically motivated employees need a working environment that offers them varied tasks, enables them to have a good relationship with their colleagues and superiors, and provides them with a safe environment. This means that employees need more room of action so that they can feel autonomous and respected which in turn positively influences motivation (Noll et al., 2017; Law and Charron, 2005; Melo et al., 2012). Furthermore, Stein and Untertrifaller (2020) found evidence in a laboratory experiment that "workers who prefer to work in an ethical work environment perform better if they are also responsible for it, compared to a situation where it was imposed on them"—underlining the effect of autonomy and responsibility on performance.

Autonomous persons are more likely to assume responsibility because they realize that they have the option to decide by themselves—to structure their life according to their well-justified choices. Psychologically empowered employees strengthen agile teams because of their acknowledged cognitive abilities that underpin the relationship of the individual to its work such as meaningfulness, competence, self-determination, and impact (Breu et al., 2002; Muduli, 2017). This aspect cannot be overemphasized, because people must be able to integrate themselves into a social system according to their abilities. Therefore, employees must experience integration as cooperation and not as oppression (cf. Boes et al., 2020a, 2020b). Psychological empowerment in turn needs to be structurally integrated to enhance resilient individuals. This is exactly where agility comes into play. An illustrative example is the fact that the more supervised employees are, the less likely they are to trust other individuals (Grund and Harbring, 2009). However, mutual trust allows for a more respectful and equal treatment of each other which in turn fosters a work culture that supports cooperative behavior and thus needs to rely less on contractual or institutionalized procedures (Mulki et al., 2006; Lester and Brower, 2003; Chow et al., 2009).

Figure 2 shows the relationship between these concepts. If developers are structurally empowered their self-perception as respected autonomous beings and the feeling of responsibility increase their motivation (Valentine and Fleischman, 2008; Koronios et al., 2019). This is a crucial cornerstone of ethical deliberation. In order to motivate people to look for ways to build normative desirable software products, we must provide the opportunity to deliberate, decide and execute the code based on that justified decision. In fact, if a software developer has no other choice but to implement software in a specifically predetermined way, we exclude the developer from the process of designing ethically sound software and have to rely on upper hierarchies to simply get the product normatively correct from the start. Hence, it will be more likely that ethical issues will only be recognized ex post. This, in turn, will either lead to them being ignored or require costly changes to alter the product after completion (Beck and Andres, 2004, Ch. 13; Brey, 2012).

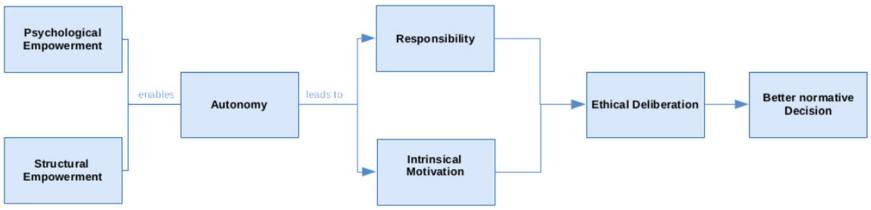

Fig. 2 Empowerment, autonomy, and ethical deliberation.

Ethical deliberation needs functioning teams and participation. The third aspect why agile methods lend themselves to be enriched with ethical deliberation is the emphasis that is put on the value of teamwork and the constant sharing of knowledge. This is of great importance when we want to embed a procedure that is based on rationally discursive elements to formulate, prioritize, and decide upon values and their technical implementation. Additionally, the procedural approach will promote ethical habits within the software community, i.e., establish critical thinking and value explorations as normal every-day behavior with regard to designing and developing digital products. This is precisely what Vallor (2016) points to when she discusses virtues as the foundation of ethically sound software systems. Virtues form an *ethos* that stabilizes and structures good normative software engineering skills. This results in the creation of an environment in which developers not only think in terms of technical functionality but also in terms of ethically desirable features and products.

In agile processes, teams are given a high degree of autonomy, being referred to as "self-organized teams" in Scrum and "empowered teams" in extreme programming. As a best-case scenario, teams with high maturity and experience can independently organize their work and choose which resources should be applied to what part of the development task and at what time. Work experience and working with peers have also been empirically connected to positively influence ethical decision making (Craft, 2013; Chow et al., 2009; Flynn and Wiltermuth, 2010). An agile team needs to have the ability to understand and implement the items in the backlog which often leads to the fact that a team is to be composed of several people with different backgrounds and skills. The fact that the team has a certain degree of autonomy regarding the planning of the work requires constant communication between team members so that every single team member has to have at least a basic understanding of the skills of their teammates as well as the knowledge of what they are working on and how this integrates with their own work. Knowledge silos are thus less likely to emerge. Ethical deliberation also relies on communication among team members and the disruption of knowledge silos facilitates identifying ethical issues that emerge through the interaction of different parts of software. The obvious advantage in agile methods lies in the fact that teamwork is already at the core of the methodology and thus does not need to be artificially introduced, e.g., as a special ethics meeting between developers who otherwise do not share their work or have no personal connections. Interestingly, the strengthening of teamwork through the process of mutual discussions during ethical deliberations could also, in turn, provide a positive effect on the promotion of agility among developers (Mudili, 2017).

Agile processes promote techno-ethical realism. Due to the iterative approach and the constant increments of software, ethical deliberations tend to be less abstract because reasoning is more concrete and object-focused: Since the goal of each sprint is a "working prototype" certain practical consequences of the design are already tangible during the development process. As outlined above, within the agile process incremental releases are continuously delivered so that unsatisfactory elements due to technological problems or in regard to usability can be detected early on and changed. Thus, many difficulties in handling and operation can be solved immediately. It also ensures that any extensive and rigid requirements set at the beginning of the project are constantly adjusted, weighed or, if necessary, dropped in order to be able to develop an adequate product. This can also be understood as a conservative stance within technology assessment methodologies—a stance that puts more focus on generic normative issues rather than on speculative ones (Brey, 2012), i.e., big data refers to privacy issues: Due to the amount of data required, big data inevitably leads to ethical and legal deliberations in dealing with privacy, be it data storage and the corresponding questions of data security and accessibility or how this data has to be collected. Privacy is therefore a generic ethical issue. The de facto handling of existing software guides further thinking and specifies abstract ideas. Ethical deliberations benefit from this because normative concerns can be localized and responded to more directly through this step-by-step development. This, in turn, supports reasoning on normative questions at each level of abstraction without going astray, i.e., technological thinking about war drones does not entail general moral questions on the issue of a "just war" which is located at a political level. Normative expectations thus become tangible and less speculative, as they are aligned with concrete objects. Requirements that have not been met are therefore made visible, and requirements that do no longer seem reasonable because they leave the considered domain can be identified and rejected. In this sense, the ethical object-orientation emphasizes a techno-ethical realism.

Sprints offer a salient endpoint to deliberation. Another advantage that agile processes offer are the clearly structured time frames that are tied to specific deliverables ("working prototype at the end of the sprint"). Since normative deliberations, such as factual knowledge, can be inexhaustible, people tend to find themselves in a position in which they search for the perfect normative design or, if the deliberation is on an issue close to their heart, focus on the perfect implementation of every single detail. The psychological burden that one "has not done enough" might be an impediment that hinders the continuation of development.

While the statement that trade-offs matter in software design as they do in life is certainly trivial, it is still important to keep in mind that ethical deliberation is itself subject to empirical constraints. Among these are a realistic account of human (psychological) limitations and the economic costs of a (prolonged) ethical deliberation, namely in the form of search costs. It seems obvious that it is not sensible to spend a long time deliberating on small details with little impact just to aim for a maximization of the normative good design. Of course, the higher the stakes, the more time should be spent deliberating—but a final decision needs to be made eventually. Time and the associated costs are a limiting factor in every software development process and the time attributed to the deliberation of ethical issues should be spent wisely. Obviously, any deliberation that

claims to produce any practical impact must (eventually) come to an end. Between the two extremes—an omniscient being who does not need to deliberate at all and imperfect beings with limited knowledge such as humans, who could deliberate endlessly—there is a sweet spot, the perfect balance between advantages of deliberating about how to implement normative features into software and the costs of prolonging deliberation (time to market, etc.). In a way this mirrors the debate in psychology and economics between the two concepts of "maximizing" and "satisficing" which dates back to Simon (1956). If deliberation is a process "that begins in ignorance and ends in knowledge" (Johnson, 2007) and given the fact that a complete and maximizing ethical deliberation is not realistic, we must look for a reasonable "end point" at which the deliberation stops. The *concept of satisficing* calls this the "aspiration level." This level serves as a stopping rule once a certain level of "good enough" is achieved. Here, choosing the first alternative that satisfies an aspiration level, e.g., the considered value has been given proper and "good enough" consideration, instead of a continued deliberation process might be a more feasible solution.

Agile processes with their concrete focus on time frames offer clear endpoints at which deliberation must come to an end and a decision must be made: either stop deliberating when any issues seems sufficiently explored for the time being or, if deliberation needs to continue, actively (and intentionally) push it forward to the next sprint. Especially with regard to normative issues and the assumption of responsibility resulting from recorded and documented decisions, external guidelines help to reach a decision when it is due. In a sense this feature of agile processes lies within the domain of moral design (Gigerenzer, 2010), which focuses on the creation of suitable environments for ethical deliberation to prosper rather than to focus on the values of any single developer.

Concluding remarks

In this paper, we argued that agile development processes can include and enhance ethical deliberation due to certain features of this particular organizational style—among them the focus on autonomy of the engineer promoted by flat hierarchies, the existing focus on team collaboration, the provision of techno-ethical realism, and the introduction of clear endpoints for (ethical) deliberations.

While it is a promising idea that a widespread development style can foster ethical deliberation during the development process, it is also necessary to think about how, when, and who must tackle these questions within management (Umbrello and Gambelin, 2022). It is also necessary to think about the possibility of formalizing ethical procedures regarding generic as well structural ethical topics and implement them within the agile framework.This calls for further empirical research which needs to test the implementation of ethical deliberation into agile, e.g. which scrum ceremonies are particularly apt to foster deliberations, to monitor design choices and to increase feedback with stakeholders. Furthermore, ethically informed engineering has to entail continuous observation and possibly alteration of digital artifacts throughout their entire life cycle focusing on normative good design via DevOps.

Received: 19 October 2021; Accepted: 16 May 2022;
Published online: 06 June 2022

References

Abbas AE, Senges M, Howard RA (2019) A Hippocratic Oath for Technologists. Next-Generation Ethics: Engineering a Better Society, 71 in: Abbas AE (ed) Nextgenerationethics: Engineering a better society. Cambridge University Press, pp. 71–81

Agar N (2019) How to be human in the digital economy. MIT Press

Anderson C (ed) (2017) Overcoming challenges to infusing ethics into the development of engineers: Proceedings of a workshop. National Academies Press

Beck K, Andres C (2004) Extreme Programming Explained: Embrace Change Second Edition. *XPSer*

Boes A, Gül K, Kämpf T, Lühr T (2020) Empowerment als Schlüssel für die agile Arbeitswelt. In: Gestaltung vernetzt-flexibler Arbeit. Springer Vieweg, Berlin, Heidelberg, pp. 89–102

Boes A, Gül K, Kämpf T, Lühr T (eds) (2020) Empowerment in der agilen Arbeitswelt: Analysen, Handlungsorientierungen und Erfolgsfaktoren. Haufe-Lexware

Breu K, Hemingway CJ, Strathern M, Bridger D (2002) Workforce agility: the new employee strategy for the knowledge economy. J Inf Technol 17(1):21–31

Brey P (2000a) Disclosive computer ethics. ACM Sigcas Comput Soc 30(4):10–16

Brey P (2000b) Method in computer ethics: towards a multi-level interdisciplinary approach. Eth Inf Technol 2(2):125–129

Brey PA (2012) Anticipatory ethics for emerging technologies. Nanoethics 6(1):1–1

Briand L, Bianculli D, Nejati S, Pastore F, Sabetzadeh M (2017) The case for context-driven software engineering research: generalizability is overrated. IEEE Softw 34(5):72–75

Cao L, Ramesh B (2008) Agile requirements engineering practices: an empirical study. IEEE Softw 25(1):60–67

Cho J (2008) Issues and challenges of agile software development with SCRUM. Issues Inf Syst 9(2):188–195

Chow WS, Wu JP, Chan AK (2009) The effects of environmental factors on the behavior of Chinese managers in the information age in China. J Bus Eth 89(4):629–639

Coeckelbergh M (2021) AI for climate: freedom, justice, and other ethical and political challenges. AI Eth 1(1):67–72

Craft JL(2013) A review of the empirical ethicaldecision-making literature: 2004-2011. J bus ethics 117(2):221–259

Dancy J (2004) Ethics without principles. Oxford University Press on Demand

Dancy J (2018) Practical shape: a theory of practical reasoning. Oxford University Press

Digital.ai (2020). 14th Annual State of Agile Report. State of Agile https://explore.digital.ai/state-of-agile/14th-annual-state-of-agile-report

Dikert K, Paasivaara M, Lassenius C (2016) Challenges and success factors for large-scale agile transformations: a systematic literature review. J Syst Softw 119:87–108

Ebert C, Gallardo G, Hernantes J, Serrano N (2016) DevOps. IEEE Softw 33(3):94–100

Flanagan M, Howe DC, Nissenbaum H (2008) Embodying values in technology: theory and practice. Inf Technol Moral Philos 322:24

Flanagan M, Nissenbaum H (2014) Values at play in digital games. MIT Press

Flewelling P (2018) The Agile Developer's Handbook: get more value from your software development: get the best out of the Agile methodology. Packt Publishing Ltd

Floridi L (2008) Foundations of information ethics in: Himma KE, Tavani HT (Eds) The handbook of information and computer ethics. John Wiley & Sons, pp. 3–25

Floridi L (2011) A defence of constructionism: philosophy as conceptual engineering. Metaphilosophy 42(3):282–304

Flynn FJ, Wiltermuth SS (2010) Who's with me? Falseconsensus, brokerage,and ethical decision making in organizations. Acad Manage J 53(5):1074–1089

Fogg BJ (2002) Persuasive technology: using computers to change what we think and do. Ubiquity 2002:2

Förster K, Wendler R (2012) Theorien und Konzepte zu Agilität in Organisationen. Dresdner Beiträge zur Wirtschaftsinformatik, NR. 63/12

Friedman B, Nissenbaum H (1997) Software agents and user autonomy. In Proceedings of the first international conference on Autonomous agents, 466-469

Friedman B, Kahn P, Borning A (2002) Value sensitive design: theory and methods. University of Wahington technical report, pp. 2–12

Friedman B, Kahn PH, Borning A (2008) Value sensitive design and information systems in: Himma KE, Tavani HT (eds) The handbook of information and computer ethics. John Wiley & Sons, pp. 69–101

Friedman B, Hendry DG (2019) Value sensitive design: shaping technology with moral imagination. MIT Press

Gablas B, Ruzicky E, Ondrouchova M (2018) The change in Management Style during the course of a project from classical to the agile approach. J Competitiveness 10(4):38–53

Gigerenzer G (2010) Moral satisficing: rethinking moral behavior as bounded rationality. Top Cogn Sci 2(3):528–554

Gogoll J, Zuber N, Kacianka S, Greger T, Pretschner A, Nida-Rümelin J (2021) Ethics in the software development process: from Codes of Conduct to Ethical Deliberation. Philos Technol 1–24

Grund C, Christine H (2013) Trust and control at the workplace: Evidence from representative samples of employees in Europe. J Econ Stat (Jahrbuecher fuer Nationaloekonomie und Statistik), 233(5–6) 619-637

Grunwald A (2015) Technology assessment and Design for Values. Handbook of ethics, values, and technological design, pp. 67–86

Grunwald A (2018) Technology assessment in practice and theory. Routledge
Harris T (2016) How technology Hijacks people's minds—from a magician and Google's design ethicist. Medium Mag. https://medium.com/thrive-global/how-technology-hijacks-peoples-minds-from-amagician-and-google-s-design-ethicist-56d62ef5edf3
Hewlett Packard (2017) Agile is the new normal: adopting Agile project management. Hewlett Packard Enterprise Development LP
Hubig C (2015) Die Kunst des Möglichen I. Transcript-Verlag
IEEE (2020) Draft standard for model process for addressing ethical concerns during system design 7000TM/D3. IEEE
Ijsselsteijn W, De Kort Y, Midden C, Eggen B, Van Den Hoven E (2006) Persuasive technology for human well-being: setting the scene. In: International conference on persuasive technology. Springer, Berlin, Heidelberg, pp. 1–5
Johnson RN (2007) Prichard, Falk, and the end of deliberation. Can J Philos 37(Suppl1):131–147
Judy KH (2009) Agile principles and ethical conduct. In: 2009 42nd Hawaii international conference on system sciences. IEEE, pp. 1–8
Kidd PT (1994) Agile manufacturing: forging new frontiers. Addition-Wesley, England
King J, McCrudden CJ (2017) The Dark Side of Nudging: The Ethics, Political Economy, and Law of Libertarian Paternalism. Hart Publishing
Koning T, Koot W (2019) Agile Transformation: KPMG Survey on Agility. KPMG. Retrieved from https://assets.kpmg/content/dam/kpmg/nl/pdf/2019/advisory/agile-transformation.pdf
Koronios K, Kriemadis A, Dimitropoulos P, Papadopoulos A (2019) A values framework for measuring the influence of ethics and motivation regarding the performance of employees. Bus Entrep J 8(1):1–19
Law A, Charron R (2005) Effects of agile practices on social factors. In: Proceedings of the 2005 workshop on human and social factors of software engineering, pp. 1–5
Lester SW, Brower HH (2003) In the eyes of the beholder: the relationship between subordinates' felt trustworthiness and their work attitudes and behaviors. J Leadersh Organ Stud 10(2):17–33
Macintyre A (1981) After virtue. University Press of Notre Dame, Notre Dame
Manders-Huits N, Zimmer M (2009) Values and pragmatic action: The challenges of introducingethical intelligence in technical design communities. Int Rev Inf Ethics 10, 37–44
Mathur A, Acar G, Friedman MJ, Lucherini E, Mayer J, Chetty M, Narayanan A (2019) Dark patterns at scale: findings from a crawl of 11K shopping websites. Proc ACM Hum-Comput Interact 3(CSCW):1–32
Mathur A, Kshirsagar M, Mayer J (2021) What makes a dark pattern… dark? Design attributes, normative considerations, and measurement methods. In: Proceedings of the 2021 CHI conference on human factors in computing systems. pp. 1–18
McLennan S, Fiske A, Celi LA, Müller R, Harder J, Ritt K, Haddadin S, Buyx A (2020) An embedded ethics approach for AI development. Nat Mach Intell 2:488–490
Melo CDO, Santana C, Kon F (2012) Developers motivation in agile teams. In: 2012 38th Euromicro conference on software engineering and advanced applications. IEEE, pp. 376–383
Metcalf J, Moss E, Boyd D (2019) Owning ethics: corporate logics, Silicon Valley, and the institutionalization of ethics. Soc Res 82:449–476
Mittelstadt B (2019) Principles alone cannot guarantee ethical AI. Nat Mach Intell 1(11):501–507
Moor JH (2005) Why we need better ethics for emerging technologies. Eth Inf Technol 7(3):111–119
Muduli A (2017) Workforce agility: examining the role of organizational practices and psychological empowerment. Global Bus Organ Excell 36(5):46–56
Mulki JP, Jaramillo F, Locander WB (2006) Emotional exhaustion and organizational deviance: can the right job and a leader's style make a difference? J Bus Res 59(12):1222–1230
Nida-Rümelin J (2000) Rationality: Coherence and Structure in: Nida-Rümelin J, Spohn W (eds) Rationality, Rules, and Structure. Theory and Decision Library, vol 28. Springer, Dordrecht, pp. 1–16
Nida-Rümelin J (2005) Angewandte Ethik Die Bereichsethiken Und Ihre Theoretische Fundierung: Ein Handbuch, Kröner Verlag, 2. Auflage
Nida-Rümelin J (2009) Philosophie und Lebensform. Suhrkamp, Frankfurt am Main, pp. 14–15
Nida-Rümelin J, Weidenfeld N (2018) Digitaler Humanismus: Eine Ethik für das Zeitalter der Künstlichen Intelligenz. Piper, München
Nida-Rümelin J (2019) Structural rationality and other essays on practical reason, vol. 52. Springer
Nida-Rümelin J, Gutwald R, Zuber N (2021) Structural rationality. In: Knauff M, Spohn W (eds) The handbook of rationality. The MIT Press, pp. 625–633
Nida-Rümelin J (2020) Eine Theorie praktischer Vernunft. De Gruyter, Berlin
Nissenbaum H (1999) Can Trust be Secured Online? A theoretical perspective. Etica E Politica, 1(2)
Noll J, Beecham S, Razzak A, Richardson B, Barcomb A, Richardson I (2017) Motivation and Autonomy in Global Software Development. In: Oshri I, Kotlarsky J, Willcocks L (eds) Global Sourcing of Digital Services: Micro and Macro Perspectives. Global Sourcing 2017. Lecture Notes in Business Information Processing, vol 306. Springer, Cham, pp. 19–38
Open Data Institute. Data skills framework https://theodi.org/article/data-skills-framework/. Accessed 29 Mar 2022
Ott K (2005) Technikethik in: Nida-Rümelin J (ed) Angewandte Ethik Die Bereichsethiken und Ihre Theoretische Fundierung: Ein Handbuch, Kröner Verlag, 2. Auflage, pp. 568-648
Özdemir Ş (2020) Digital nudges and dark patterns: the angels and the archfiends of digital communication. Digit Scholarsh Humanit 35(2):417–428
Palm E, Hansson SO (2006) The case for ethical technology assessment (eTA). Technol Forecast Soc Change 73(5):543–558
Poth A, Jacobsen J, Riel A (2020) Systematic Agile Development in Regulated Environments. In: Yilmaz M, Niemann J, Clarke P, Messnarz R (eds) Systems, Software and Services Process Improvement. EuroSPI 2020. Communications in Computer and Information Science, vol 1251. Springer, Cham, pp. 191–202
Ramos C, Augusto JC, Shapiro D (2008) Ambient intelligence—the next step for artificial intelligence. IEEE Intell Syst 23(2):15–18
Reijers W, Coeckelbergh M (2020) A narrative theory of technology. In: Narrative and technology ethics. Palgrave Macmillan, Cham, pp. 79–111
Reitberger W, Ham J, Weiss A, Spahn A, Meschtscherjakov A, Nickel P, Tscheligi M (2009) The ubiquitous persuader: mechanisms, applications and ethical dilemmas of ambient persuasion. In: 3rd European Conference on Ambient Intelligence (AmI 2009). International Ambient Media Association (iAMEA), pp. 201–207
Richards D, Dignum V (2019) Supporting and challenging learners through pedagogical agents: addressing ethical issues through designing for values. Br J Educ Technol 50(6):2885–2901
Rohbeck J (1993) Technologische Urteilskraftzu einer Ethik technischen Handelns, Suhrkamp Verlag
Rottig D, Koufteros X, Umphress E (2011) Formal infrastructure and ethical decision making: an empirical investigation and implications for supply management. Decis Sci 42(1):163–204
Russo F (2012) The Homo Poieticus and the Bridge Between Physis and Techne . In: Demir H (eds) Luciano Floridi's Philosophy of Technology. Philosophy of Engineering and Technology, vol 8. Springer, Dordrecht, pp. 65–81
Ruijters E, Stoelinga M (2015) Fault tree analysis: a survey of the state-of-the-art in modeling, analysis and tools. Comput Sci Rev 15:29–62
Savolainen J, Kuusela J, Vilavaara A (2010) Transition to agile development-rediscovery of important requirements engineering practices. In: 2010 18th IEEE international requirements engineering conference. IEEE, pp. 289–294
Sarkis J (2001) Benchmarking for agility. Benchmarking: Int J 8(2):88–107. https://doi.org/10.1108/14635770110389816
Schneier B (1999) Attack trees. Dobb's J 24(12):21–29
Schwaber K, Beedle M (2002) Agile software development with Scrum, vol 1. Prentice Hall, Upper Saddle River
Schwaber K, Sutherland J (2011) The scrum guide. Scrum Alliance 21:19
Senges M, Ryan PS, Whitt RS (2017) Composite ethical frameworks for IoT and other emerging technologies. Available at SSRN 3092362
Shen H, Deng WH, Chattopadhyay A, Wu ZS, Wang X, Zhu H (2021) Value cards: an educational toolkit for teaching social impacts of machine learning through deliberation. In: Proceedings of the 2021 ACM conference on fairness, accountability, and transparency. pp. 850–861
Shilton K (2013) Values levers: building ethics into design. Sci Technol Hum Values 38(3):374–397
Shim W, Lee SW (2017) An agile approach for managing requirements to improve learning and adaptability. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). IEEE, pp. 435–438
Simon HA (1956) Rational choice and the structure of the environment. Psychol Rev 63(2):129
Simon J (2016) Values in design. In: Heesen J (ed) Handbuch Medien-und Informationsethik. Springer-Verlag
Spreitzer GM (2008) Taking stock: a review of more than twenty years of research on empowerment at work. Handb Organ Behav 1:54–72
Stahl BC, Flick C (2011) ETICA workshop on computer ethics: exploring normative issues. In: Fischer-Hübner S, Duquenoy P, Hansen M, Leenes R, Zhang G (eds) Privacy and identity management for life. Privacy and identity 2010. IFIP advances in information and communication technology, vol 352. Springer, Berlin, Heidelberg
Staffelbach B, Arnold A, Feierabend A (2014) Fehlverhalten und Courage am Arbeitsplatz–analysiert anhand des Schweizer HR-Barometers. Diskussionspapiere der Tagungdes Ausschusses "Wirtschaftswissenschaften und Ethik" des Vereins für Socialpolitik, Frankfurt am Main
Steed R, Caliskan A (2021) Image representations learned with unsupervised pretraining contain human-like biases. In: Proceedings of the 2021 ACM conference on fairness, accountability, and transparency. pp. 701–713
Stein C, Untertrifaller A (2020) The effect of ethical responsibility on performance (No. 99176). University Library of Munich, Germany

Spiekermann S (2015) Ethical IT innovation: A value-based system design approach. CRC Press
Spiekermann S, Winkler T (2020) Value-based engineering for ethics by design. arXiv preprint arXiv: 2004.13676
Solinski A, Petersen K (2016) Prioritizing agile benefits and limitations in relation to practice usage. Softw Qual J 24(2):447–482
Sverrisdottir HS, Ingason HT, Jonasson HI (2014) The role of the product owner in scrum-comparison between theory and practices. Procedia-Soc Behav Sci 119:257–267
Tariq S, Jan FA, Ahmad MS (2016) Green employee empowerment: a systematic literature review on state-of-art in green human resource management. Qual Quant 50(1):237–269
Tavani HT (2013) Ethics and technology: controversies, questions, and strategies for ethical computing. Wiley, Hoboken, NJ
Turk D, France R, Rumpe B (2002) Limitations of agile software processes. In: Third international conference on eXtreme programming and agile processes in software engineering (XP 2002). pp. 43–46
Umbrello S, Gambelin O (2022) Agile as a vehicle for values: a value sensitive design toolkit. In: Santa-Maria Andres, Fritzsche Albrecht (eds) Philosophy of engineering and technology. Springer, Cham, (in press)
Vakkuri V, Kemell KK, Jantunen M, Halme E, Abrahamsson P (2021) ECCOLA—a method for implementing ethically aligned AI systems. J Syst Softw 182:111067
Valentine S, Fleischman G (2008) Ethics programs, perceived corporate social responsibility and job satisfaction. J Bus Eth 77(2):159–172
Vallor S (2016) Technology and the virtues: a philosophical guide to a future worth wanting. Oxford University Press
Van Bruchem-Visser RL, van Dijk G, de Beaufort I, Mattace-Raso F (2020) Ethical frameworks for complex medical decision making in older patients: a narrative review. Arch Gerontol Geriatr 90:104160
Van Den Eede Y (2011) In between us: on the transparency and opacity of technological mediation. Found Sci 16(2–3):139–159
Van den Hoven, J., Vermaas, P. E., & Van de Poel I. (2015) Design for values: Anintroduction in: Van den Hoven, J., Vermaas, P. E., & Van de Poel I. (eds) Handbook of ethics, values, and technological design: Sources, theory, values and application domains. Dordrecht: Springer Netherlands, pp.1–7
Van Lamsweerde A (2001) Goal-oriented requirements engineering: a guided tour. In: Proceedings of the fifth IEEE International symposium on requirements engineering. IEEE, pp. 249–262
Van Oyen MP, Gel EG, Hopp WJ (2001) Performance opportunity for workforce agility in collaborative and noncollaborative work systems. IIE Trans 33(9):761–777
Van Wynsberghe AL, Moura GM (2013) The concept of embedded values and the example of Internet Security. Responsible research and innovation in ICT, Oxford, Technical Report, 1101
Van Wynsberghe A (2021) Sustainable AI: AI for sustainability and the sustainability of AI. AI Eth 1:213–218
Verein Deutscher Ingenieure (VDI) (2000) Technology assessment concepts and foundations. VDI 3780
Zuber N, Kacianka S, Pretschner A, Nida-Rümelin J (2020) Ethical deliberation for agile processes: the EDAP manual. In: Hengstschläger M (ed) Digital transformation and ethics. ecowin, pp. 134–150
Zwart SD (2014) Modeling in design for values. In: Van den Hoven J, Vermaas P, van de Poel I(eds) Handbook of ethics, values, and technological design. Springer, Dordrecht

Funding

Open Access funding enabled and organized by Projekt DEAL.

Competing interests

The authors declare no competing interests.

Ethical approval

Not applicable.

Informed consent

Not applicable.

Additional information

Correspondence and requests for materials should be addressed to Jan Gogoll.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.