Do You Own a Volkswagen? Values as Non-Functional Requirements

Balbir S. Barn

Do You Own a Volkswagen? Values as Non-Functional Requirements

Balbir S. Barn

Middlesex University, Hendon, London, UK, NW4 4BT

Abstract. Of late, there has been renewed interest in determining the role and relative importance of (moral) values in the design of software and its acceptance. Events such as the Snowden revelations and the more recent case of the Volkswagen "defeat device" software have further emphasised the importance of values and ethics in general. This paper posits a view that values accompanied by an appropriate framework derived from non-functional requirements can be used by designers and developers as means for discourse of ethical concerns of the design of software. The position is based on the Volkswagen "Dieselgate" case study and a qualitative analysis of developers views from Reddit discussion forums. The paper proposes an extension of an existing classification of requirements to include value concerns.

1 Introduction

Values are a key driver of human behaviour and are seen as an important component in societal and environmental sustainability. However, current software engineering practice does not pay sufficient attention to the notion of values given the role of software and systems in so-called "smart city" sustainability actions. Critically, there is no sufficiently expressive machinery to describe how values such privacy and security are elicited, negotiated, mediated and accommodated in systems design beyond either early stages such as contextual design. For the purposes of this paper, values are what Friedman refers to: ownership and property; privacy, freedom from bias, universal usability, trust, autonomy, informed consent and identity. She defines values as : what a person or group of people consider important in life [13].

The paper's position is motivated by the recent so-called "Dieselgate" media story concerning Volkswagen. On 18th September 2015, Volkswagen, USA was accused by the US Environment Protection Agency (EPA) of installing an illegal "defeat device" software that dramatically reduces nitrogen oxide (NOx) emissions - but only when the cars were undergoing strict emission tests.

This paper advances the notion that (moral) values are distinct from non functional requirements or even *softgoals* as in goal oriented requirements engineering (GORE) approaches but can benefit from being incorporated into Non-Functional Requirements (NFR) frameworks. In order to present this position, the paper takes a case example approach. Firstly, a summary outlining the key

issues arising from the Volkswagen "defeat device" is presented in section 2. The case example is analysed with respect to value concerns. In contrast to other recent analyses of the Volkswagen case [29], the paper augments the analysis by taking concrete views from developers. Evidence is drawn from the Reddit discussion forum section on Coding where developers posted 80 items over a four day period [25]. Section 3 presents this analysis. In doing so, this paper concretely proposes how an existing classification mechanism for requirements can be extended. Section 4 draws on the academic literature on values, ethics and non functional requirements to provide further support for the proposition that values can be classified as a special case of non-functional requirement. In section 5, the paper concludes by offering a roadmap for further research.

2 Case Study and Approach

Interpretative, exploratory case study research is a first step to understanding phenomena as input for further research. The Volkswagen case study is important from two perspectives: Firstly, its impact on environmental pollution and secondly, the subsequent apportioning of blame on software engineers responsible for the "defeat device" software and their (implicit) unethical actions. A descriptive (non-value laden) account of the Volkswagen case study is presented next using the investigative reports from two quality newspapers, the Guardian and the New York Times.

2.1 Volkswagen case study

The story was first reported in the media on the 18th of September, 2015 with a claim that around 482,000 cars were affected in USA. However this story begins much earlier. In 2014, a non-profit group, the International Council on Clean Transportation worked with the California Air Resources Board and researchers at West Virginia University to conduct on-road diesel emissions tests of cars including two Volkswagens and a BMW X5. The researchers found that when tested on the road some cars emitted almost 40 times the permitted levels of nitrogen oxides yet performed flawlessly in lab conditions. Subsequently the federal body and the Environmental Pollution Agency (EPA) were both involved in negotiations with Volkswagen. Eventually in September 2015, when delays on announcing new 2016 models could not be explained, the EPA announced the violation of the Clean Air Act [33]. The basis of this violation was the discovery of software sub-routines that detected when the car was being tested and then activated equipment that reduced emissions. But the software turned the equipment off during regular driving, increasing emissions far above legal limits, possibly to save fuel or to improve the car's torque and acceleration. Note that while all cars undergo strict emission tests, manufacturers have always been able to manipulate the settings of the parameters for emission tests. The story developed and up to 11 million cars were affected [26] leading to around 25 billion Euro being wiped off the company's market value [3]. Eventually, the CEO,

Martin Winterkorn resigned but denied personal wrongdoing. At the same time. VW's CEO in USA, Michael Horn in a testimony before US Congress identified the true authors of his company's deception [7]: "This was a couple of rogue software engineers who put this in for whatever reason."

As a further justification, he described the "defeat device" as a line of code "hidden in millions of software code". Since then, following investigations in Germany, other senior managers appear to have been implicated and were suspended. Huber, the acting head of VW's supervisory board, called the crisis a "moral and policy disaster" and went on to say: "The unlawful behaviour of engineers and technicians involved in engine development shocked Volkswagen just as much as it shocked the public" [1].

Since initial reporting of this case, Federal and California regulators have begun an investigation into a second computer program in Volkswagen's diesel cars that also affects the operation of the cars' emission controls.

2.2 Approach

A notable aspect of the media reporting of the Volkswagen defeat device story is the lack of discussion of ethical considerations that engineers are subject to, in the act of engineering a system. Both US and Germany have strong engineering codes of ethics yet these have not featured in any reporting. Given the relationship between ethics and values, this paper analyses this case study from that perspective and by drawing upon a qualitative analysis of the Reddit discussion forum section on coding. The thread, entitled: " 'Rouge'(sic) Software Engineers blamed for VW emissions (would this be a bug or feature?)" received 80 items that were posted over a over four day day period (12 Oct to 15 Oct)[25].

The approach presented has reliability limitations of the analysis of the Reddit posts. A single developer forum was identified through search queries. Various forums such such as StackOverflow were investigated but there were no discussions about the Volkswagen "defeat device". There are a limited number of participants who posted discussion items. Although there are over 65,000 readers who subscribe to the coding forum, 27 participants posted comments, from which 7 posted at least twice. The data is is largely indicative and an appropriate future research avenue would be to explore developer's perspectives.

3 Case Study Analysis

The first question is one of ethics and its relationship to (software) engineering and the section begin with a provocative prompt:

"Engineers are the un-acknowledged philosophers of the postmodern world." [21].

In part, the reason for this is straightforward: an engineer may not ask about what they should be doing, or may find that a solution that cannot be achieved by technical expertise alone. From this it follows that ethical judgements such as responses to questions of environmental protection following violations of the

Clean Air Act are required. Hence, philosophy is an internal practical need. So did the "rogue" engineers alluded to by Horn seek this ethical judgement? A more complex explanation is also offered by Mitcham: engineering is *modelling* a new philosophy of life. One interpretation of this is: we have moved from a natural environment to one where we increasingly present humans as information-centric entities whose production and consumption of information occurs through engineered artefacts. Such a world demands continuous evaluation of ethical concerns.

The codes of engineering ethics such as that developed by the Association of German Engineers [8] [1] provide some insight as to what is expected in terms of norms of behaviour from engineers. For example:

- Engineers are responsible for their professional actions and tasks corresponding to their competencies and qualifications while carrying both individual and shared responsibilities.
- Engineers are aware of the embeddedness of technical systems into their societal, economic and ecological context, and their impact on the lives of future generations.
- Engineers apply to their professional institutions in cases of conflicts concerning engineering ethics.

The Engineering ethics principles also give guidelines on how to resolve conflicting values. In cases of conflicting values, engineers give priority: to the values of humanity over the dynamics of nature; to issues of human rights over technology implementation and exploitation; to public welfare over private interests; and to safety and security over functionality and profitability of their technical solutions. It would appear that the software developers responsible for developing and installing the "defeat device" violated a normative code of conduct. What is not known is whether the developers concerned were able to escalate it to their managers, their professional body and in the last resort, directly inform the public or refuse co-operation altogether as directed in paragraph 3.4 of the VDI code of ethics for engineers [8]. Both the media reporting and the posts on the Reddit Coding forum by developers find no mention of ethical concerns directly. For the latter, there are references and extensive dialogue that tries to account for defensive actions and how to minimise potential fallout:

gullibleboy: I agree. Thankfully, I have never been asked to write code that was blatantly illegal. And I would like to think I would have enough integrity to say no. But, I certainly would make sure to get written confirmation, from my superiors, to ensure that I was not made the patsy. Any seasoned developer would do the same.

Retaining evidence of written confirmation such as in email, while potentially difficult and desirable raises other legal issues.

LongUsername: Will be surprising to see if they have written evidence. It happened at least 5 years ago, and I know many companies with an automatic email deletion policy of about 2 years. Even if there were explicit instructions, unless the engineer thought enough to archive the email off (in violation of data retention policies), it may be gone.

[1] `http://www.vdi.eu/engineering/`

Resolving conflicting values appears to be core philosophical action that engineers will always have to deal with. While the VDI code of ethics provides rules to apply in cases of conflict, it may not always be appropriate to recourse to the professional body. Some times conflict of values are also a proxy for conflicting goals. The comment by *frezik* below makes that point. It might be the case that the requirements specifications are poorly specified and the design process does not allow for the conflict resolution. It could also be possible that "requirements creep", created un-wanted changes that should not have got through an approval process.

mallardtheduck: In a "normal" vehicle, all profiles should conform to the relevant emissions standards. In this case, my suggestion is that VW's programmers/engineers started with "good" profiles and over the process of tweaking/improving them, ended up making the more common profiles violate said standards. Whether this was done deliberately or not is hard to say. Either way it's something that should have been caught and probably was, but was "hushed up" by persons higher up the chain.

The idea of a "rogue" software engineer is also interesting when observed from different perspectives. A "rogue" engineer could have inserted the software that did exactly what the company required against the express wishes of her manager. In which case, the problem is even bigger in terms of questioning the entire software engineering practice at Volkswagen.

PCLoad_Letter: From this you can draw the conclusion that there are no software audits on the code running millions of these cars and no one in QA or the engineering divisions even questioned when the emissions tests came back much lower that expected? That is a much bigger problem than a Rogue engineer.

Conversely, in his testimony to Congress, Michael Horn, although implausible, may have meant to imply that the engineers were "rogue" because they had not reported the software behaviour as required by the engineering codes of practice and their company. A notion gently supported by a developer:

MuonManLaserJab: For it to be "rogue engineers", they would have had to decide to keep the facts a secret while writing the cheat...

It is constructive to consider why the developers on Reddit have focussed on defence rather than ethical concerns. Partly, it is explained by how engineers work. The way that engineering (and code development) happens as reported in [21] by Louis Bucciarelli in his ethnographic studies of engineers: when students are doing engineering problems it is generally thought that they "*ought not to get bogged down in useless 'philosophical' diversions*" [6, pp. 105-6]. At this stage of their engineering career, students are practicing to be engineers. The developer's world (when they are engaged in problem solving) is also highly abstract. It is one of programming syntax, data structures, and program comments that refer only to functional requirements. Such an abstract world does not leave room for ethical concerns that might have serious implications for society at large. The question arises: where and how in the process of design can these critical diversions be accommodated? The problem is further compounded by deadline pressures resulting in poor decisions and is also offered as an explanation.

frezik: The situation I'm thinking of here is when the programmer is told "we need the ECU tweaked on this car to make x mpg highway while passing emissions tests, and we need it by date y". They make a profile that makes the mpg requirement, and they make

another profile that makes the emissions requirement. They try to combine them to meet both goals, but they run out of time. Under pressure, they do something stupid and illegal, and make the emissions test run when test mode is detected, and mpg mode otherwise.

Following on from ethical concerns, is a consideration of who else was involved the chain of decision making that led to the insertion of the "defeat device" software. The developers point out that software running the engine control unit would have gone through numerous code reviews.

rfinger1337: VW has a responsibility to know whats in the code. It wouldn't get past testing, code review...

GuyNamedNate2: It really depends on what their code reviewing /auditing practices are...I would hope and expect an organization as big as VW would have avionics-quality processes.

AllGloryToHypno-Toad: if this company is writing engine management software without code reviews and testing, then that's another issue. This should have come up and many, many times.

Modern software engineering can also create spaces whereby an engineer may not be aware that he has violated his engineering ethics codes. For example, component based design mandates black box design through well defined interfaces that supports specific functional requirements. Developers may end up writing code for a software component without an awareness of the wider context of where that component may be used. Thus we see:

LongUsername: Unfortunately, The real coders may not have known what they were doing. Tell the programmer who does the emissions system to "disable the emissions system when this flag over here is set... It's for test-purposes only." and tell the guy working on the steering system "when the wheel doesn't change position for X amount of time, set this global flag over there that says we're on a Dyno".

Cultural issues within the organisation are also of concern. The engineering community either chose deliberately to not report the rogue engineers to the organisation or even outside, or were not able to because of the cultural climate. A Reuters report published in the Guardian on 10th October described the culture under the former CEO, Martin Winterkorn [1] as authoritarian. Bernard Osteloh was quoted as saying: "We need in future a climate in which problems aren't hidden but can be openly communicated to superiors...We need a culture in which it's possible and permissible to argue with your superior about the best way to go.". The reality as presented by the developers is different:

*_Toranaga_*: I dunno, fall guys get fired... Whistle-blowers get exiled to Siberia.

4 Values versus Non Functional Requirements

In this section, it becomes apparent that a common implicit position in current software engineering practice is that of non functional requirements (NFR) being used as as a proxy for accounting for values. The literature is examined and used to argue that existing approaches for managing NFRs would not have sufficed to prevent the Volkswagen case. A further argument advanced is that values need to be unpicked in order for them to be used effectively by engineers. Simply, we want to provide engineers with the necessary machinery to allow them to externalise their philosophical deliberations as they practice their craft.

4.1 Non Functional Requirements

NFRs have been extensively studied by the software engineering community as they are recognised as important factors to the success of a software project [12, for example]. Despite their importance, NFRs are poorly understood and often neglected in the software design process. A satisfactory understanding of NFRs remains elusive and is attributed to three key problems: lack of a workable definition; difficulties of classifying types of NFRs; and representing NFRs in the design process [15]. Further, once NFRs have been identified, they need to be assigned properties (attributes) which can be measured. Not all NFRs have attributes and notably, the systematic review by Mairiza et al shows that of the 252 types of NFR identified, over 50% were without attributes [18]. This is relevant to values.

One consistent view of NFRs is: they are a *quality* attribute of a system that its stakeholders care about and hence will affect their degree of satisfaction with the system. These qualities end up being referred to as *"ilities"*. Examples include: reliability, testability, usability, portability etc. Mairiza et al. identify 115 such quality characteristics [18]. Such sayilities are hard to characterise in ways that can support rigorous engineering and are ultimately subjective and stakeholder specific [11]. Further, implementations of sayilities may be cross-cutting across many software components and may also impinge on the meta-systems used for constructing a system under question.

An inspection of several empirical studies all omit any consideration of treating values as NFRs [19,18,31,16]. While security and privacy are both moral values, their treatment as NFRs are from a system perspective and not from that of the end user's moral concerns. A survey of software architects to consider NFRs conducted by Ameller et al. also found that values as we define them do not feature as NFRs even in a category of non-technical NFRs (i.e. not those such as performance etc.) [2]. Overall, this implies a conclusion where, either values are systematically ignored in the practice of NFR elicitation or values may not be NFRs. Certainly, the ethics/values discussions did not feature strongly in the media reporting and the coder's comments on Reddit.

We now consider the relevance of techniques of requirements engineering to values. Over recent years, goal oriented requirements engineering (GORE) [22] has began to dominate practice [16] particularly with respect to NFRs with the NFR Framework [9] being the most widely cited. In the NFR Framework, NFRs or goals that are hard to express (*softgoals*), and their decisions are captured in goal graphs and refined into detailed concrete goals. Others followingthe GORE tradition include i* [35]. GORE based techniques present a variety of options for analysis such as providing a more formal basis of how goals realise other goals, conflict between goals and the positive and negative contributions goals make to other goals and ultimately tradeoffs between goals. GORE approaches would appear to be a promising area for further examination from a values perspective.

4.2 Values and Value Sensitive Concerns

Investigating the intertwining and entailment of values and technology development has been an ongoing area of research mostly originating in the domain of human-computer interaction (HCI) [30]. In the HCI literature, values are identifiable entities that are built in by design or accident through the affordances of the technology [13]. These considerations have been termed Value Sensitive Design (VSD) by Friedman [13], who refers to: ownership and property; privacy, freedom from bias, universal usability, trust, autonomy, informed consent and identity.

This is contrasted with sociology and social psychology where values are criteria that are used to evaluate or make judgements about events or people encountered, helping explain individual and collective behaviour [5]. As social interactions become increasingly mediated by technology then these two interpretations of values merge and in doing so, they govern user action or non-action within technology [17]. Thus design choices that explicitly consider values can change the affordances of resulting technologies [14,27].

The problem of identifying and qualifying values has many facets: their individual or shared nature; their realisation in concrete features; their subjective or objective location; their accidental or intended role with respect to system functionality [28]. A further complexity is that all these facets operate along a continuum so an intended or accidental property is not simply two possible states of a value. These complexities prevent their widespread acknowledgement and treatment. In particular, there is no general theoretical framework to allow stakeholders in the design process to identify, qualify and successively map into usable data the relevant values for the application at stake. In short, the understanding of values and their contribution to technology acceptance is not theoretically grounded into the entire software development lifecycle. As discussed earlier, GORE approaches (particularly, the use of softgoals) have the potential to represent values. However, softgoals cannot be directly linked to direct or indirect stakeholders. Further, "'Values are not goals, they are assumptions (more precisely, evaluations). A value is a judgment, though very general and vague. It says of something that it is good or bad. A goal is a regulatory state in someone's mind" ([20] reported by Pommeranz et al. [23]). It would appear that values and their multi-dimensional nature requires new GORE-like approaches that allow designers to engage in a philosophical discourse about their systems and their shaping of society.

4.3 Value Architectures

In summary, values are not referenced in meta reviews of NFRs, nor are example values listed as an "ility". So either values are systematically ignored in the practice of NFR elicitation or values may not be NFRs. The complexity of values cannot be accounted for by simple attributes and importantly, typical NFR elicitation approaches do not allow designers to engage in a necessary philosophical

dialogue about those "*...problems that engineers admit cannot be resolved simply with engineering methods alone....professional ethical issues.*" [21, p.31].

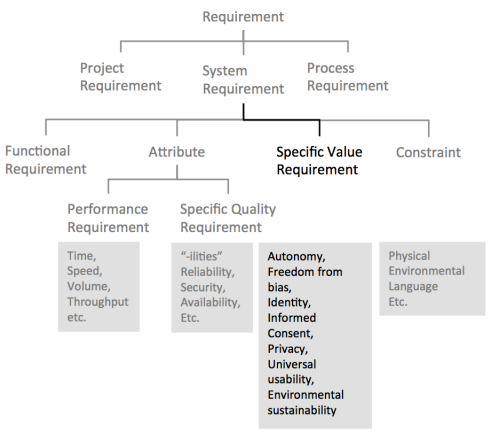
It is also clear that GORE based approaches, particularly the notion of *softgoals*, is a promising area of research for incorporating values as these provide qualitative mechanisms for resolution that are sufficient. Thus we propose the following:

1. Extend the Glinz classification to include a values category;
2. Develop processes and the necessary vocabularies to allow ethical discussions of the impact of values of design decisions that can contribute a *"value architecture"* in much the same way as functional requirements contribute to *application* architectures and NFRs lead to *technical* architectures of system.

Glinz proposed a classification of requirements based on concerns - something that is a matter of interest to a system. A concern is a performance concern if timing or volume is a matter of interest. The set of all requirements are partitioned into functional requirements, performance requirement, specific quality requirements and constraints. Alongside the classification, are a set of rules that are applied in order in order to classify a given requirement. We propose extending this classification and rule base as follows.

We introduce *Value* defined as: a requirement that pertains to value concern. Such a concern is classified by the introduction of an additional classification rule. "...a specific moral value that the system or component shall either support or prevent erosion of.". The resulting classification and rule set is shown in figure 1. The benefits of this extension allows us to specifically consider values as a concern in the requirements process. While it is relatively straight forward to develop measures for attributes such as Performance, defining a measure for value is much harder and requires agreement amongst stakeholders. By interacting with stakeholders using the trigger question, we open a channel for that necessary philosophical dialogue alluded to by Mitcham and necessary for the agreement.

Having addressed the requirements classification concern, the next stage is to develop value sensitive processes that can take advantage of the classification. The proposal here is to adapt the work of Yoo et al [34] to support value elicitation, ethical discussions of the impact of values of design decisions and their capture suitable for engineers. Their work develops a framework for accounting for values using participatory design approaches. Evaluations on the efficacy of this approach have been reported elsewhere [4] but the technical machinery to support this remains an ongoing research challenge. For example, can a value expression language be used to both specify value requirements, and provide input for technology acceptance models such [32]? Such models use survey based research instruments that do not have questions about values. Practical considerations can be dealt with as functional requirement specifications are developed and be part of the descriptive frameworks used in requirement specification. A greater challenge is during the design of the software interactions. It is proposed that the trigger question can be integrated in processes that check if a requirement has been implemented. With regard to the the original case study, stages



No.	Question: Was this requirement stated because we need to specify...	Result
1	...some of system's behaviour, data, input, or reaction to input stimuli ? regardless of the way how this is done?	Functional
2	... restrictions about timing, processing or reaction speed, data volume, or throughput?	Performance
3	... a specific quality that the system or a component shall have?	Specific quality
4	...a specific moral value that the system or component shall either support or prevent erosion of.	Specific value
5	... any other restriction about what the system shall do, how it shall do it, or any prescribed solution or solution element?	Constraint

Fig. 1. Extension of Glinz's concern-based taxonomy and classification rules

where the trigger question could have been invoked include: code reviews, and componentisation of software.

5 Conclusion

The Volkswagen case study illustrates the importance of the role of ethics and its subsuming notion of values in software practice. Developers currently do not appear to engage in meaningful discussions about values that are explicit or those that emerge through the use of systems. It is notable, that the developers on the Reddit platform were relatively subdued in their discussion of ethical / value concerns. This paper has proposed that such hesitancy is partly linked to how functional and non functional requirements are managed. On the surface, values appear to have similarities with NFRs but the view is taken that just as NFRs present problems of representation, definition and classification, values may generate similar issues. Hence an extension of Glinz's concerns-based taxonomy for requirements that accounts for values is presented. In doing so, a channel for necessary dialogue with stakeholders about the ethics of decisions that programmers and designers is opened. Epistemological study to develop the necessary frameworks for such a dialogue is an important first step. Future research plans include: empirical data collection of values and their importance to relevant communities such as developers; the development of informal, semi-formal and formal conceptual models of values and related concepts; and experimental evidence of evaluation of such models. Identifying where trigger questions can be incorporated is also a critical element in any processes. In today's hyper-connected world where systems are increasingly delivered in a pervasive form, such as through "apps" on smart phones, the risk to erosion of values, as well as, threats to social and societal sustainability are paramount. Values need to be formally accounted for in software engineering practice.

References

1. Reuters Agency. Volkswagen executives describe authoritarian culture under former ceo. `http://www.theguardian.com/business/2015/oct/10/volkswagen-executives-martin-winterkorn-company-culture`, 2015.
2. David Ameller, Claudia Ayala, Jordi Cabot, and Xavier Franch. How do software architects consider non-functional requirements: An exploratory study. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 41–50. IEEE, 2012.
3. George Arnett. The scale of the volkswagen crisis in charts. `http://www.theguardian.com/news/datablog/2015/sep/22/scale-of-volkswagen-crisis-in-charts`, 2015.
4. Balbir Barn and Ravinder Barn. Resilience and values: Antecedents for effective co-design of information systems. In *23rd European Conference on Information Systems (ECIS 2015), AISNet Library*, 2015.
5. Roger Bennett. Factors underlying the inclination to donate to particular types of charity. *International Journal of Nonprofit and Voluntary Sector Marketing*, 8(1):12–29, 2003.
6. Louis L Bucciarelli. *Designing engineers*. MIT press, 1994.
7. C-SPAN.org. Hearing on volkswagen emissions violations. `http://www.c-span.org/video/?328599-1/hearing-volkswagen-emissions-violations`, 2015.
8. H Christ. Fundamentals of engineering ethics. *VDI the Association of Engineers, Dusseldorf*, 2002.
9. Lawrence Chung, B Nixon, E Yu, and J Mylopoulos. Non-functional requirements. *Software Engineering*, 2000.
10. A. Dardenne, A. Van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of computer programming*, 20(1-2):3–50, 1993.
11. Ke Dou, Xi Wang, Chong Tang, Adam Ross, and Kevin Sullivan. An evolutionary theory-systems approach to a science of the ilities. *Procedia Computer Science*, 44:433–442, 2015.
12. Christof Ebert. Putting requirement management into praxis: dealing with non-functional requirements. *Information and Software technology*, 40(3):175–185, 1998.
13. Batya Friedman. Value-sensitive design. *Interactions*, 3(6):16–23, 1996.
14. Batya Friedman and Helen Nissenbaum. Bias in computer systems. *ACM Transactions on Information Systems (TOIS)*, 14(3):330–347, 1996.
15. Martin Glinz. On non-functional requirements. In *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*, pages 21–26. IEEE, 2007.
16. Aldrin Jaramillo Franco. Requirements elicitation approaches: A systematic review. In *Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on*, pages 520–521. IEEE, 2015.
17. Edwin A Locke. The motivation sequence, the motivation hub, and the motivation core. *Organizational behavior and human decision processes*, 50(2):288–299, 1991.
18. Dewi Mairiza, Didar Zowghi, and Nurie Nurmuliani. An investigation into the notion of non-functional requirements. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 311–317. ACM, 2010.
19. Abderrahman Matoussi and Régine Laleau. A survey of non-functional requirements in software development process. *Departement dÕInformatique Universite Paris*, 12, 2008.

20. Maria Miceli and Cristiano Castelfranchi. A cognitive approach to values. *Journal for the Theory of Social Behaviour*, 19(2):169–193, 1989.
21. Carl Mitcham. The importance of philosophy to engineering. *Teorema: Revista Internacional de Filosofía*, pages 27–47, 1998.
22. J. Mylopoulos, L. Chung, and E. Yu. From object-oriented to goal-oriented requirements analysis. *Communications of the ACM*, 42(1):31–37, 1999.
23. Alina Pommeranz, Christian Detweiler, Pascal Wiggers, and Catholijn Jonker. Elicitation of situated values: need for tools to help stakeholders and designers to reflect and communicate. *Ethics and Information Technology*, 14(4):285–303, 2012.
24. Eric S Raymond. *The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary.* " O'Reilly Media, Inc.", 2001.
25. reddit. "rouge" software engineers blamed for vw emissions (would this be a bug or feature?). `https://www.reddit.com/r/coding/comments/3ogtqw/rouge_software_engineers_blamed_for_vw_emissions/?`, 2015.
26. Graham Ruddick. Vw scandal: chief executive martin winterkorn refuses to quit. `http://www.theguardian.com/business/2015/sep/22/vw-scandal-escalates-volkswagen-11m-vehicles-involved`, 2015.
27. Katie Shilton. Values levers: Building ethics into design. *Science, Technology & Human Values*, 2012.
28. Katie Shilton, Jes A Koepfler, and Kenneth R Fleischmann. Charting sociotechnical dimensions of values for design research. *The Information Society*, 29(5):259–271, 2013.
29. Diomidis Spinellis. Developer, debug thyself. *Software, IEEE*, 33(1):3–5, 2016.
30. Lucy Suchman. Do categories have politics? the language/action perspective reconsidered. In *Human values and the design of computer technology*, pages 91–106. Center for the Study of Language and Information, 1997.
31. Richard Berntsson Svensson, Martin Höst, and Björn Regnell. Managing quality requirements: A systematic review. In *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*, pages 261–268. IEEE, 2010.
32. Viswanath Venkatesh, Michael G Morris, Gordon B Davis, and Fred D Davis. User acceptance of information technology: Toward a unified view. *MIS quarterly*, pages 425–478, 2003.
33. Wikipedia. Clean air act (united states). `https://en.wikipedia.org/wiki/Clean_Air_Act_(United_States)`, 2015.
34. Daisy Yoo, Alina Huldtgren, Jill Palzkill Woelfer, David G Hendry, and Batya Friedman. A value sensitive action-reflection model: evolving a co-design space with stakeholder and designer prompts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 419–428. ACM, 2013.
35. E.S.K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 226–235. IEEE, 1997.