Editor: **Tim Menzies**
North Carolina State University
tim@menzies.us

Is Your Software Valueless?

Jon Whittle

SOFTWARE DEVELOPMENT IGNORES human values. As a society, we rely on software systems that neither align with nor respect our core values, such as transparency, gender diversity, social justice, and personal integrity. The past 50 years of software engineering have focused on functionality, cost, safety, availability, and security. But what about broader human values (Figure 1) such as compassion, social responsibility, and justice? The way we design software fundamentally influences society, yet human values—which we would all claim to care about—have been a side concern in software engineering. (See "Where Are the Values in Software?")

Digital Object Identifier 10.1109/MS.2019.2897397
Date of publication: 16 April 2019

Surely it is high time that we fundamentally reimagine the way we design software. Rather than focusing only on a narrow set of concerns, we should embed all human values into software design. If we don't, we, as software engineers, will inadvertently create a society that nobody wants.

If such a system was designed with the value of gender equality in mind, the interface would be designed very differently.

Admittedly, embedding human values into software is difficult. Even where there is a willingness, and managerial support, to think about values, it proves challenging. Software development defaults back to values that are relatively easy to deal with—accessibility, usability, and availability—for which there are more clearly defined guidelines or tools. Thus, running tools on a graphical user interface that check whether a color scheme is readable by the color-blind population is common but does not help with addressing broader human values. There is a mismatch between what the software development community values—typically automation, productivity, and quality—and broader societal values (Figure 2). There is also often an assumption that the latter naturally leads to the former, that we are improving people's lives by automating things using quality software.

Why do values in software matter? The issue of values, and more narrowly ethics, in computing is receiving renewed attention because there are doomsday predictions of discriminatory artificially intelligent systems taking over the world. However, the real problems are much more mundane than are those of sentient machines. The interface for a simple human resources (HR) recruitment system is one example. Most job interviewers are encouraged, if not mandated, to consider periods of time away from work, e.g., parental leave. Yet HR recruitment systems do not prominently display such information; to discover it, interviewers must search through potentially hundreds of curriculum vitaes. If such a system was designed with the value of gender equality in mind, the interface would be designed very differently. We have known for decades that design is not values

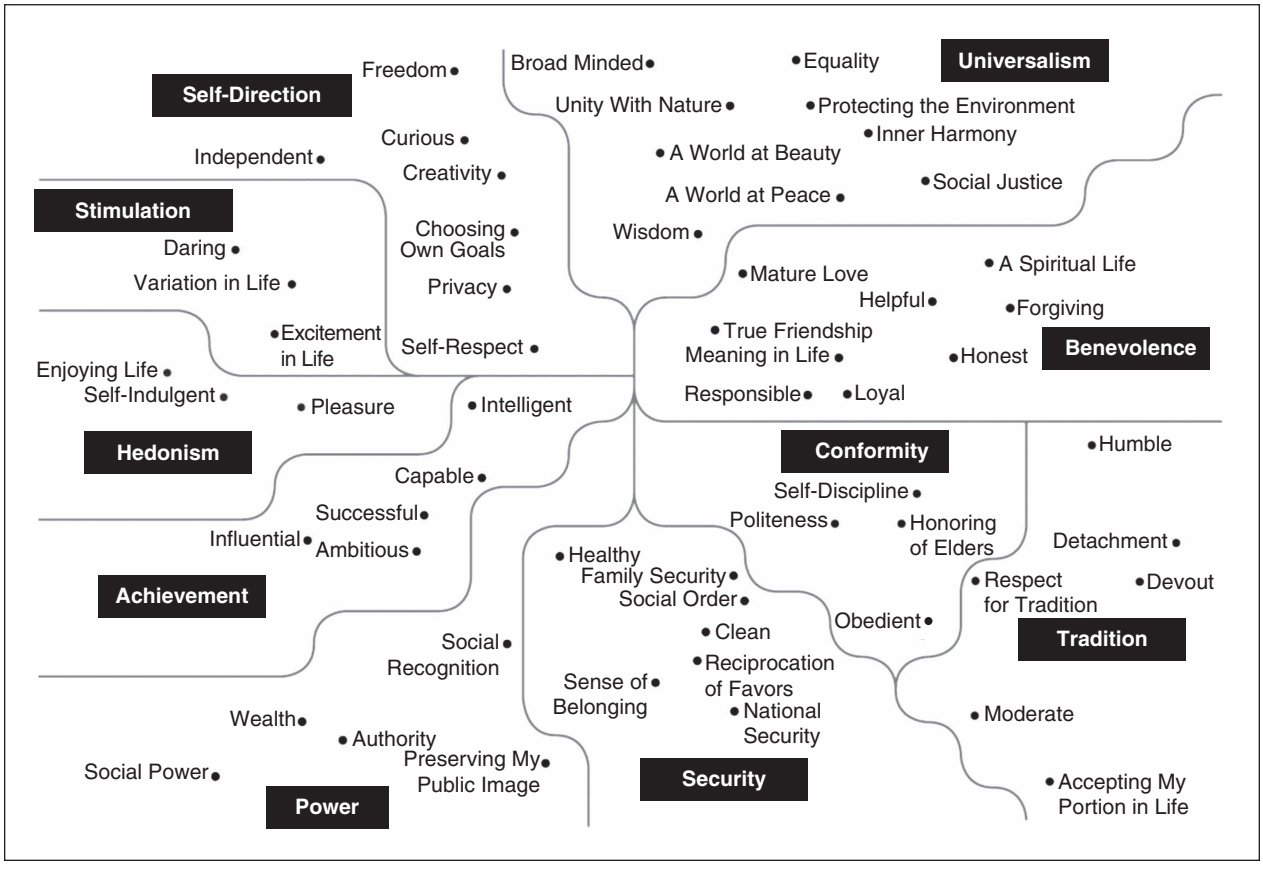

FIGURE 1. A number of theories have attempted to define values, such as the Schwartz Theory of Basic Values.[1] The 10 universal values are boxed and subdivided into finer-grained values. Values closer to each other (as defined by the bullet points) are complementary, and those farther apart are in conflict. Note that values are different from ethics: ethics are culturally agreed-upon moral principles, while values make no moral judgment, so, for example, making money is a perfectly acceptable value. (Adapted from Schwartz[8].)

neutral, but software engineers have failed to understand this. We examined the last four years of papers published in the top software engineering conferences and journals; we found that only 16% of papers considered values at all. Of those, a significant majority focused on values of security and privacy.

But does the software industry care about nonfinancial values? A naive view might argue that it does not, that it is all about the bottom line. However, companies have at least claimed to care about values for a long time. Ever since Jim Collins' and Jerry Porras' book *Built to Last*,[2] which found that a key determinant of a company's success is a strong values statement, organizations have put a lot of effort into defining their corporate values. A study by Maitland[3] found that 86 of Financial Times Stock Exchange 100 Index companies have public values statements, with values such as corporate integrity, respect, and honesty topping the list. Clearly, companies implement these with various levels of seriousness, but many, if not all, do take them seriously and have managerial mechanisms to create a values culture. However, there is no way to trickle these values down into the software that we build.

At Monash University, Melbourne, Australia, we recently conducted two case studies with software companies to improve our understanding of their approaches to values. At least for these companies, values are explicitly talked about during software development: this usually takes the form

of a values document used hiring decisions, training new staff, performance appraisals, and strategic decision making. Sometimes there are more sophisticated ways to create an open culture, ones in which software developers can talk about and honestly question company values. These include forums for stepping back from day-to-day concerns, having someone designated as a critical friend, clarity in hiring practices, and training programs. However, even in those cases in which values are considered, the approach is limited to creating a values-driven culture as opposed to having it engrained into the act of software development. When companies consider values in developing software, it is during business analysis and requirements engineering only; values are easily forgotten later.

Therefore, the current state, at least for software companies that appreciate values, is a reliance on organizational culture. However, there is very little, if anything, to support technical work. This is not all bad news, however. Many existing software development approaches can be adapted easily to work with values. Therefore, while a revolution in the mind-set of developers is necessary, more of an evolution could suffice from a process point of view. For example, agile development methods lend themselves naturally to thinking about values. In their current form, values will not immediately receive attention, but through a designated values guardian in a Scrum team, user stories could easily become values stories, and measurement approximations, such as T-shirt sizing, would be useful for dealing with the inherent complexity of a nuanced concept such as values. Furthermore, we would go a long way by introducing well-established participatory design techniques into user-experience and user-centered design approaches. Participatory design methods are good at ensuring that end-user values are taken into account but have suffered from a lack of clarity and a refusal to simplify.[5]

More generally, there could be values versions of successful tricks that the software industry has used. Imagine a values manifesto with the beauty and simplicity of the agile

WHERE ARE THE VALUES IN SOFTWARE?

Value is an overloaded term. Values-based methods are well known in human–computer interaction (HCI) and information systems[4,9,10] but are nonexistent in software engineering. HCI and information systems do not deal with the business of actually building software, so although they could apply in the early stage of software engineering, they offer little guidance as to how to handle values in the more technical stages of development. The word *value* is often referred to in agile methods, but then the focus is only on business value. Similarly, Boehm's value-based software engineering[11] deals almost entirely with economic value. Some emerging works in software engineering take a more human-values approach, such as GenderMag[12] for discovering gender bias in software, but this is still very early.

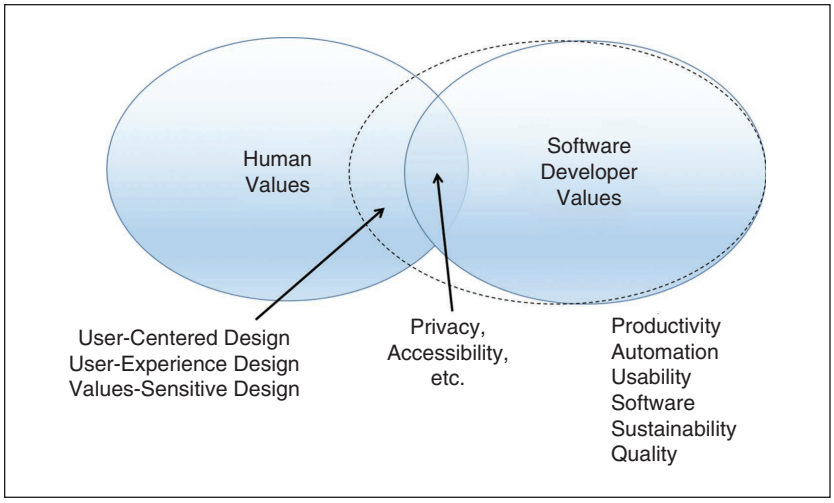

FIGURE 2. A software developer's values versus human values: software engineers value technical concepts such as productivity, automation, usability, and quality, with the assumption that these traits naturally lead to broader human values. This is a naive view, however; human values and software developer values rarely coexist. Methods such as user-centered design, user-experience design, and values-sensitive design[4] take a broader view, but we are a long way from full alignment between human and developer values.

manifesto, making it clear to developers that values are important. Or there could be a values maturity model that helps organizations to self-assess their values culture, such as level 0, ad-hoc consideration of values; level 1, a clear, published corporate values statement but no real way to implement it; level 2, some processes to deal with values; level 3, proactive and structured ways to ensure that values are considered at all stages of the software lifecycle; and level 4, software tools to support values. You get the idea. Requirements engineering methods could be applied easily to refine what values mean. After all, one of the biggest challenges in instilling values in software is that values are, by definition, vague concepts. However, specifying values in concrete terms, in the context of an actual project, plays to the strengths of requirements engineering methods. Also, well-accepted technical methods could be adapted to look at software development through a values lens: take A/B testing, for example, which could be used to test out how different software versions impact values.

It is time that the software industry takes human values seriously, but not just for the greater social good. Violations of human values can have serious negative financial consequences for the economy. In a sample of Internet security breaches, Cavusoglu[6] found an average market-capitalization loss of US$1.65 billion for the companies affected. In the Volkswagen (VW) emissions scandal,[7] software designers deliberately contradicted the company's corporate value of responsible thinking, a decision that led to the resignation of the chief executive officer, a 30% drop in VW's stock price, and a 25% drop in sales within one year. Therefore, value violations are big business. Software researchers and practitioners must respond by doing what they do best: creating methods for handling such problems before a catastrophe hits.

Acknowledgments

The ideas in this article benefited from discussions with Waqar Hussain and Davoud Mougouei.

References

1. S. Schwartz, "Basic human values: Theory, measurement, and applications," *Revue Francaise de Sociologie*, vol. 47, no. 4, pp. 929–985, 2006.
2. J. Collins and J. Porras, *Built to Last: Successful Habits of Visionary Companies*. New York: HarperBusiness, 1997.
3. S. Walker, "The values most valued by UK plc," Maitland. Accessed on: Feb. 20, 2019. [Online]. Available: http://www.maitland.co.uk/wp-content/uploads/2015/10/20151001-Maitland-Values-Report.pdf
4. B. Friedman, D. Hendry, and A. Borning, "A survey of value sensitive design methods," in *Foundations and Trends in Human Computer Interaction*. Boston: Now Publishers, 2017.
5. J. Whittle, "How much participation is enough? A comparison of six participatory design projects in terms of outcomes," in *Proc. Participatory Design Conf.*, 2014, pp. 121–130.
6. H. Cavusoglu, B. Mishra, and S. Raghunathan, "The effect of Internet security breach announcements on market value: Capital market reactions for breached firms and Internet security developers," *Int. J. Electron. Commerce*, vol. 9, no. 1, pp. 70–104, 2004.
7. B. Georgievski and A. Alqudah, "The effect of the Volkswagen scandal: A comparative case study," *Res. J. Finance Accounting*, vol. 7, no. 2, pp. 54–57, 2016.
8. S. Schwartz, "I've built a good mousetrap and people come to use it," *Psychologist*, vol. 31, pp. 56–59, June 2018.
9. C. Knobel and G. Bowker, "Values in design," *Commun. ACM*, vol. 54, no. 7, pp. 26–28, 2011.
10. G. Cockton, "Value-centred HCI," in *Proc. NordCHI*, 2004, pp. 149–160.
11. S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grunbacher, *Value-Based Software Engineering*. New York: Springer, 2006.
12. M. Burnett et al., "GenderMag: A method for evaluating software's gender inclusiveness," *Interacting Computers*, vol. 28, no. 6, pp. 760–787, 2016.

ABOUT THE AUTHOR

JON WHITTLE is the dean of the faculty of information technology at Monash University. Contact him at Jon.Whittle@monash.edu.