

It is not all straw, but it can catch fire: In defense of impossible ideals in computing¹

Chuck Huff
St. Olaf College

James Stieb proposes a profitable misreading of Donald Gotterbarn's approach to responsible computing, claiming that the position ultimately leads to holding computer professionals "responsible for an undisclosed, perhaps unlimited, list of 'undesirable events'" [1] and that the practical problems associated with any implementation of this approach are insurmountable. It is easy to show that this critique is a straw man and slightly more difficult to show that many of the supposedly insurmountable difficulties are likewise made of straw.

Still, this misreading is profitable for two reasons. First it is the most likely misreading one will find within particular organizations dedicated narrowly to financial success. Understanding how this misreading emerges within a moral ecology [2] can help one to sympathize with Stieb's position and help prepare students for careers that may include these moral ecologies.

Second, when one looks closely at the values of the computing profession it becomes clear that there is more than straw in the accusation that they are impossible ideals. All moral claims have a "characteristic of the ultimate" [3] that makes them difficult to completely satisfy. But, contrary to the accusation that this is a flaw, I will argue here that these impossible ideals are essential to our human nature [4] and to the nature of good computing – they are part of what makes the committed professional catch fire and strive for excellence.

To begin, a quick review of Stieb's critique: Stieb complains that the "open-ended" nature of positive responsibility is pernicious in that its hyperinflation of responsibility can result in holding "every engineer morally 'responsible' for all bad effects to mankind" [1]. Any attempt to limit the hyperinflation raises "a significant epistemic problem" – in a world of conflicting claims about the good, whose account of the good will be accepted? As a result, Stieb claims, this unattainable ideal "may cause cynicism among those in the field" [1].

Michael Davis [5] reports the results of a large empirical investigation of communication between engineers and managers in firms that had significant investment in engineering (e.g. automotive, chemical, technology, aerospace). The role that engineers played in these organizations depended crucially upon what my colleagues and I [2] call "moral ecology," the somewhat stable, but constantly negotiated set of values, practices, and influences within societies, organizations, professions, and work groups. *Finance-driven* organizations give priority to maximizing financial goals, and engineers in these firms do not participate in decision making, but instead stand outside the process in a consultant-like role, providing information, producing design options, and answering questions. In

¹ Huff, C. (2008). "It is not all straw, but it can catch fire: In defense of impossible ideals in computing - A comment on "A Critique of Positive Responsibility in Computing"." *Science and Engineering Ethics* **14**(2): 241-244.

this environment, engineering criteria are devalued compared to management goals, and the engineer's ethical responsibilities are limited to exercising *due care* when providing advice and services, avoiding *conflicts of interest*, and remaining loyal to the legitimate interests and objectives of managers [6]. In finance-driven moral ecologies, professional obligations such as "avoid harm to others" and "contribute to society" (from the Association for Computing Machinery/Institute of Electrical and Electronics Engineers joint code of ethics) must all be phrased in terms of the engineer's "due care" and are thus inevitably cast as defensive, blame-avoidance suggestions. This produces the classic blame-avoidance memo documenting concern about a product or design strategy. In this moral ecology, the primary effect of any suggestion that computer professionals might have a positive responsibility to those other than their managers is to open them to blame when that responsibility is not fulfilled. In this light, one can see Stieb's claims as firmly grounded within this kind of organizational moral ecology (as indicated, for example, by his approving citation of Milton Friedman's defense of the profit motive alone).

There are (at least) two other moral ecologies: *quality-driven* and *customer-driven*. These organizations are constrained by financial issues, but see the achievement of either quality or superior customer service as their defining goals. In these organizations engineers participate in decision making, and managers seek consensus with engineers. Trade-offs (among cost, quality, safety, etc) are negotiated as a part of the consensus-seeking approach and managers and engineers expect to be able to come to agreement on these issues. For instance, one engineer is quoted by Davis [5, p. 133] as saying, "Cost comes in only after quality standards are met." And another offers that, "If a customer wants to take a chance, we won't go along" [5, p. 133]. In these environments, positive professional responsibilities like those that worry Stieb find a natural home.

At least some students will find themselves in finance-driven moral ecologies, and they will need to construct ways to realize their professional values in an environment that devalues their professionalism. Helping them understand this moral ecology and how communication and action work in it will be one part of a useful education in computing ethics. Answering Stieb's concern with a suite of feasible (on time and within budget) plans for action is a central part of preparing students to operate in these environments. Gotterbarn's Software Development Impact Statement (SoDIS) process is one of these feasible approaches [7].

An awareness of the variation in moral ecology convinces us that the blame problem, though important, is more a function of moral ecology than it is of positive responsibility. But how to resolve the epistemic problem Stieb cites? At some level this is an *empirical* claim: that issues of value have never historically been resolved. Stieb's citation of the philosopher Michael Foucault is an attempt to provide empirical evidence for the claim, though at a somewhat strained distance.

There are two sources of evidence more closely to hand. Michael Davis's extensive interviews with engineers in quality-driven organizations indicate that, at least in these organizations, managers and engineers *expect* to agree on how to achieve quality and

therefore spend considerable effort and resources on regularly coming to agreement [5]. This provides some evidence that agreement on these issues is possible, even in organizations that must stand the test of the marketplace.

Work that my colleagues and I have done on moral exemplars in computing constitutes another source of evidence [2, 8], though the outcome is not what one would expect: instead of widespread agreement on what is the good in computing, there was a diversity of goods in computing that different moral exemplars sought. Some were craftpersons, and designed software to help users (e.g., the handicapped) and clients (e.g., businesses who wanted to maintain customer loyalty *and* customer privacy). Others were reformers who attempted to change the field of computing (e.g., by encouraging women to participate) or to change society (e.g., by changing laws and regulation on privacy). Far from being a weakness, the diversity of visions of the good becomes a strength among these exemplars. Different professionals were spending their career energies to realize different values, all of which were worthwhile, and some of which produced useful software for a broad range of people (e.g., the predictive speller on the cell phone came from early systems to help the handicapped to type).

Still, Stieb's critique of each of these values has some power. All the exemplars realized they needed to engage in tradeoffs in their design or reform efforts, regularly coming up short in achieving the ideals (e.g., transparency, simplicity, empowerment) they had adopted. Thus each value had the open-ended character that Stieb thinks will produce cynicism. In any real world scenario, they are impossible ideals, if only because they conflict with other ideals that society would also like to maintain [9].

But contrary to Stieb's claim that they produce cynicism, these impossible ideals often produce an intense desire to improve the design, given the constraints, to better approximate the ideal. Michael Pritchard [10] gives the example of the designer of a safety belt system for window washers that an engineer spent much time perfecting, even though it already met all the minimal standards required. It was still difficult to use, and the engineer knew that safety equipment that was difficult to use was likely to hang on the scaffold unutilized.

Reinhold Niebuhr [4] makes this claim more generally about the paradox of the human condition: people have high ideals and yet are finite creatures. They desire the good and yet cannot completely achieve it. This longing is for an "impossible possibility" and is an essential part of what it means to be human. At the same time, impossible ideals are effective "tools for approximating the good" [11, p. 119].

Thus, contrary to Stieb's apprehension, there seems to be considerable evidence that impossible ideals are in fact helpful in attaining the good. In finance-driven organizations, these ideals may need to masquerade in another guise, but it is still these ideals that can set fire to the imagination of the computer professional and drive him or her to true excellence.

Acknowledgements

Thanks to Don Gotterbarn and Bill Frey for comments on earlier versions of this response. I am grateful to the National Science Foundation (DUE-9980786, DUE-9972280, and SES-0217298) for their continued support.

References

- 1 Stieb, J. (2008). A critique of positive responsibility in computing. *Science and Engineering Ethics*.
- 2 Huff, C. W., Barnard, L., & Frey, W. (In press). Good Computing: A pedagogically focused model of virtue in the practice of computing. *Journal of Information, Communication and Ethics in Society*.
- 3 Taylor, C. (1985). *Philosophy and the human sciences. Philosophical papers 2*. New York: Cambridge University Press.
- 4 Niebuhr, R. (1940). *Christianity and power politics*. New York: Scribner.
- 5 Davis, M. (1998). Ordinary technical decision making: an empirical investigation. In M. Davis, (Ed.), *Thinking like an engineer: Studies in the ethics of a profession*, (pp. 119-156). New York: Oxford University Press.
- 6 Pritchard, M. (2001). Response to 'Ordinary Reasonable Care is not the Minimum for Engineers' (Davis) *Science and Engineering Ethics* 7[2]: 291-297.
- 7 Gotterbarn, D. (December, 2001) Reducing Software Failures using Software Development Impact Statements. *Journal of Information Management Systems*.
- 8 Huff, C. W. & Rogerson, S. (September, 2005). *Craft and reform in moral exemplars in computing*. Paper presented at ETHICOMP2005 in Linköping, Sweden.
- 9 Schwartz, S. H. & Boehnke, K. (2004). Evaluating the structure of human values with confirmatory factor analysis. *Journal of Research in Personality*, 38, 230-255.
- 10 Harris, C. E., Pritchard, M, & Rabins, J. (2004). *Engineering Ethics: Case Studies*. New York: Thompson Wadsworth.
- 11 Malotky, J. D. (2003) Reinhold Niebuhr's paradox: Groundwork for social responsibility. *Journal of Religious Ethics*, 31, 101-123.