Editor in Chief: **Diomidis Spinellis**
Athens University of Economics and Business, dds@computer.org

The Social Responsibility of Software Development

Diomidis Spinellis

LIFE IN A Nazi concentration camp must have been horrendous. As an inmate, you'd almost certainly die in agony, perhaps after witnessing the tragic demise of your family and friends. As a guard, you might get to live, but with the death and suffering of millions in your conscience and your nation's collective conscience. You might think that these horrors are behind us. Think again. Humanity can advance from the unthinkable to the inevitable in small, often imperceptible, steps. You start by dismissing a Great War veteran delivering anti-Semitic polemics as an irrelevancy, you continue to ignore him when he wants to be called Führer, and you end up witnessing the industrial-scale slaughter of people.

Our world is increasingly running on software. As a software developer, for better or worse (it's up to you), you're building the fabric of our society, tomorrow's world. You're therefore directly responsible for averting some of those small imperceptible steps to future nightmares. Many of the things you do have ethical, social, and political implications.

Consider the following often-discussed examples. The filters you build in a social networking platform might create digital bubbles that reinforce a group's beliefs instead of allowing it to see others' opinions. This polarizes society and spreads hatred. Car software might protect the interests of the car manufacturer you work for, against the law, the environment, or the car's occupants. The dating site's algorithms you design might increase racial segregation and social-class divisions. (You get to play God with the human gene pool.) Your poorly designed user interface can cause people hardship, while your incorrect code can harm human life or damage property. The console game you create might be purposely designed to be highly addictive. How you route job ads can influence disadvantaged groups' work opportunities.

On a larger scale, new digital platforms, products, and services might disrupt the livelihood or working conditions of millions. Also, machine-learning algorithms might promote discrimination or produce invalid results in ways that are difficult to detect and analyze. Governments are often no better than companies. Digital government systems often operate in a void of democratic accountability and might lack sufficient recourse opportunities. The requirements and design of a system you build might subvert public policy regarding transportation, education, or law enforcement. It gets worse. The US digital mass surveillance program and discussion regarding enhanced border controls or China's evolving social-credit system might be bellwethers of a totalitarian dystopia.

From Learning to Participating

Is it realistic for you to worry about the problems I outlined? Technological progress is unstoppable and mostly welcome; thanks to it, millions around the world are healthier, wealthier, and better educated. Technology inevitably brings disruption. However, the balance between its benefits and perils is often obvious. For example, self-driving cars might displace drivers, but their widespread adoption could drastically decrease the 1.3 million deaths in road crashes each year and the 20 to 50 million people who end up injured or disabled. On the other hand, there can be no justification for developing cyberwarfare software to breach a dam's floodgates upstream of a large city. Given this complex, perilous state of affairs, what should you do? Here are some ideas.

Learn

Being a brilliant Scrum master might let you build the software right but won't ensure that you develop the right software. Read up on history to appreciate the painful and often tragic path of our society's evolution. Did you know that in the 1940s, punch card technology was used to organize Nazi concentration camps and US internment camps for Japanese Americans? Keep abreast with current events and discussion—not through 140-character tweets, but with long readings that offer insightful analysis. Acquaint yourself with the Software Engineering Code of Ethics and Professional Practice. Study subjects besides technology: philosophy, political science, sociology, ethics, and the arts. Remember: technocratic judgment can get you only so far.

Think

Contemplate the wider repercussions of what you're working on. What will happen when your shiny prototype gets widely deployed or stops being maintained? Will your new web-based service disadvantage a particular minority? What if all the data you're gathering falls into the hands of organized crime or a totalitarian government? Is the cyberweapon you're developing more likely to be used for defense or for offensive actions and terrorism that can hurt countless civilians? What if criminals can gain control of the law-enforcement backdoor access you're providing? How will your new social-interaction feature affect children or families?

Act

Walking the walk is probably the most difficult part of acting responsibly as a software developer. Strive to design and implement the software you work on so that it becomes a force for good rather than evil. Don't work for organizations whose mission is intrinsically detrimental to society. If your organization develops software that will harm society, speak up. Present your case to coworkers (sometimes engineers become blinded and seduced by the technology's power), and offer better alternatives. Don't turn whistleblower on impulse: seek alternatives, deliberate, and think through your actions. Also, each time your fancy software makes people redundant, consider the words of Hannah Arendt: "Radical evil has emerged in connection with a system in which all men have become equally superfluous."[1] Could your software instead make those people's work more meaningful?

I hear you saying that limiting your employment options or stirring up trouble in the workplace isn't realistic, when day in and day out you've got to bring bread to the table. However, the choice of where you work and what you do is rarely binary. There are thousands of choices between writing code to streamline operations at Médecins Sans Frontières (Doctors without Borders) and implementing process control software for the chemical factory that supplied sarin nerve gas to Bashar al-Assad. Depending on your personal circumstances and drive, you can certainly find ways to gravitate toward the positive end of the spectrum I outlined. By depriving

EDITORIAL STAFF

Lead Editor: Brian Brannon, bbrannon@computer.org
Content Editor: Dennis Taylor
Staff Editors: Lee Garber, Meghan O'Dell, and Rebecca Torres
Publications Coordinator: software@computer.org
Lead Designer: Jennie Zhu-Mai
Production Editor: Monette Velasco
Webmaster: Brandi Ortega
Multimedia Editor: Erica Hardison
Illustrators: Annie Jiu, Robert Stack, and Alex Torres
Cover Artist: Peter Bollinger
Director, Products & Services: Evan Butterfield
Senior Manager, Editorial Services: Robin Baldwin
Acting Editorial Content Manager: Carrie Clark
Senior Business Development Manager: Sandra Brown
Senior Advertising Coordinators: Marian Anderson, manderson@computer.org Debbie Sims, dsims@computer.org

evildoers of your talent, you contribute to everyone's well-being.

Also, if you see that your organization develops harmful software, try bringing constructive ideas to the table. You can argue that in the medium or long term, being socially responsible might benefit your organization. Or, you can come up with win–win options: software that's good for both your organization and society.

Educate

Modern software is mind-numbingly complex, and it often interacts with our world in subtle, unanticipated ways. As a software developer, you might understand these things in both the abstract and concrete, but your colleagues from other backgrounds and the public at large might not. Help by uncovering, explaining, and publicizing any issues you recognize. Build bridges with people in other domains, such as medical professionals, who might comprehend the domain's ethics but might not realize the potential, choices, and ethical issues hidden underneath the technology. Discuss with your colleagues, post in social networks, write articles, and take a stance in daily interactions. As an educated professional, you have the duty to return to society part of the education it has endowed you with. And, if you're an educator, strive to include the topics in your lectures and the curriculum.

Connect

Strengthen your voice by joining it with the voices of others. Engage with your professional societies to raise awareness of these issues and develop defense and support mechanisms so that your colleagues who speak up aren't left out in the cold. Mingle with people outside your (probably privileged) circle: those who struggle to make ends meet, those affected by your work, and scared underdogs. Write free software that could help our world become a better place. Work with think tanks, civil-society organizations, and political parties to draft and promote sensible policies. Volunteer for office and, it should go without saying, get out and vote!

Someone might argue that as software developers, we should focus just on doing our assigned work as best as we can because an organization can't function if all its members constantly question its operations. Instead, this argument goes, we should let the market decide or the government regulate what software is developed and how it's used. I disagree. This argument degrades our working existence to that of a robot. Part of being human is making moral choices in everything we do: from the detergent we buy, to the transportation we use to get to work, to the software we develop. If our children ask us in a few years what we were doing when the lights went dark, responses such as "I didn't know these horrors were happening" or "I was just doing my job" ("I was just following orders") won't cut it.

Acknowledgments

I thank Felienne Hermans, Panos Louridas, Peter G. Neumann, Nancy Pouloudi, Eberhard Wolff, and Alexios Zavras for their constructive feedback on an earlier version of this piece. The opinions expressed here are mine; no endorsement by the IEEE, its officials, or its members is implied. As always, I welcome discussion and rebuttals.

Reference

1. H. Arendt, *The Origins of Totalitarianism*, Harcourt, 1973.