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

Developer, Debug Thyself

Diomidis Spinellis

IT WAS A NEAR MISS. The press portrayed Volkswagen as the corporate villain that manipulated its diesel engines' performance during emissions tests. Yet, we software professionals know that a developer ultimately implemented the code that detected when the tests were being conducted and changed the engine's behavior accordingly. On the next scandal our profession won't escape unscathed. As professionals, we're already on the hook for developing software that regularly leaks embarrassing amounts of personal data, allows the flourishing of illegal botnets that control millions of PCs, and has provided backdoors to government agencies. Clearly, we must deal more effectively with our professional responsibility.

Continuing business as usual isn't an option. Software increasingly affects our personal lives, the economy, society, and our planet. Software engineers are daily called to participate in tricky decisions at the crossroads between ethics and more narrowly defined corporate interests. For example, consider a smartphone. Should it click when taking pictures, thereby alerting others that they're being photographed? (Mine doesn't when in silent mode.) What should it do when recording video? Should it raise the headphone audio to harmful levels? (Mine issues a warning but lets me override it.) Should it offer the option to record phone calls? How should it handle its owner's personal data? Should the engineers' allegiance be to the phone's owner or the company's shareholders? Should the phone offer a backdoor to law enforcement authorities? Should that backdoor's existence be kept secret?

Software increasingly affects our personal lives, the economy, society, and the planet.

The questions we face are difficult and will become even more so. If you think it's easy to answer the questions I outlined because a smartphone's legal owner should have some clear rights, reflect on the case of the customers using "free" social-networking applications. Also keep in mind that these questions might be more difficult to answer when a product or service is made available in totalitarian countries that regularly violate human rights. And the list of areas in which software plays a crucial role keeps growing. Apart from the well-known areas of transport, medical devices, and nuclear energy, consider that software in self-driving cars, wearables, entertainment, and the smart grid easily raises even thornier questions.

Any response dealing with software developers' professional responsibility should take into account the software's complete life cycle.

IEEE Software Mission Statement

To be the best source of reliable, useful, peer-reviewed information for leading software practitioners—the developers and managers who want to keep up with rapid technology change.

0740-7459/16/$33.00 © 2016 IEEE

Going back to the Volkswagen case, there were probably people who specified the requirement for the engine test manipulation, designed it, modeled it, constructed it, tested it, put it under configuration management, and passed it through quality assurance. Managers, development leads, and programmers allowed the mischief to happen.

> Organizations that are found to flout the rules should be forced to release their software as open source.

The Problems of Regulation

If we don't act decisively and effectively through our professional organizations, such as the IEEE Computer Society and the ACM, governments will step in with thickets of regulation. This outcome will be catastrophic to our field, ending a half century of magnificent innovation and growth that have provided countless benefits to mankind.

Excessive government regulation of the rapidly evolving field of software development will be destructive because the precise and rigid oversight of software products and processes will stifle innovation. This is already happening in many other industries, including the control of car emissions, in which targets are rigidly set through unrealistic test scenarios. Instead, some people have argued that environmental outcomes could be improved at a lower cost by addressing the problem systemically through the use of GPS and information technology to differentiate the engine parameters of cars driving in a congested city center from those of cars driving on a desert highway. These are optimizations that large and small IT companies, from Amazon to Zimbra, make daily as part of their business.

Under a tight regulation regime, we can easily imagine governments gumming up the global software industry by micromanaging product specifications and auditable deliverables between development processes. In addition, in some countries, tight regulation will breed corruption, whereas other countries will use regulation to erect new trade barriers. All the while, these countries will lose their intellectual capital as software innovation moves to those with more enlightened regulation regimes.

Regulation will also likely be ineffectual. The huge amount of software available today, coupled with the practice of end-user programming and the fact that software is woven in thin air and thus difficult to control, will leave any regulations that can be applied in practice too vague and full of loopholes.

Governments are trying to sidestep these problems by vertically regulating specific industries, as is the case for medical devices. However, this is becoming more and more difficult as software is increasingly interlinked into complex large systems of systems with fluid, permeable boundaries. Consider an implanted pacemaker that a patient can monitor through a phone app and that a physician can fine-tune remotely. What will be regulated as medical software apart from the pacemaker's code? The tuning software? The firmware on the physician's broadband router? The patient's phone app? The phone's OS? Other apps on the phone? All these entities could be maliciously used to attack the pacemaker. And if you think that strict regulation is warranted because it's better to be safe than sorry, take into account the respected voices who now argue that overregulation of drug trials often hurts patients by delaying the use of life-saving medicines.

Better Alternatives

Instead of increased government meddling, a more effective and efficient approach would be a well-functioning regime of self-regulation. Professional bodies would set broad standards of professional conduct and practice related to software development, and developers would be bound to follow them and would be responsible for their actions. Governments would oversee the regime's monitoring and enforcement.

Getting such a system to work won't be trivial. Current regimes of accreditation, certification, and licensing[1] must be adjusted to cover all areas where intervention is needed. Standardization must be extended to cover any important gaps. Software professionals' education and training must be updated to deal with these changes. Codes of ethics and professional conduct must be better communicated and enforced through actions that will have teeth. Sadly, up to now, the reaction of professional

societies regarding the engineers responsible for the Volkswagen case has been missing in action.

Transparency can also help. In code developed as open source, developers are less likely to commit shenanigans, especially if they know that from day one their contributions will be openly available, as code commits signed with their name. Furthermore, open source software exposes what's technically possible and what's not, providing the government and society with the reliable information needed to make policy decisions regarding the software's desired attributes. In contrast, proprietary software obscures true technical affordances, providing many people with a false sense of security while only marginally inconveniencing criminals. Examples of this phenomenon include ostensibly secure systems of lawful interception and digital rights management.

Transparency can also help motivate compliance: organizations that are found to flout the rules should be forced to release their software as open source. This would be a powerful deterrent, while providing the research community and professional societies with material we can use to improve our understanding of compliance monitoring and enforcement.

The risks of misbehaving software have been with us for decades but are now becoming too ubiquitous to casually brush under the carpet. We must act now; otherwise, the next software scandal might take down software development as we know it.

Reference

1. P. Kruchten, "Licensing Software Engineers?," *IEEE Software*, vol. 25, no. 6, 2008, pp. 35–37.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.

We welcome your letters.

Send them to software@computer.org.
Include your full name, title, affiliation, and email address.
Letters are edited for clarity and space.

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 Specialist: Mark Bartosik
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
Manager, Editorial Services Content Development: Richard Park
Senior Business Development Manager: Sandra Brown
Senior Advertising Coordinators: Marian Anderson, manderson@computer.org
Debbie Sims, dsims@computer.org

CS PUBLICATIONS BOARD

Jean-Luc Gaudiot (VP for Publications), Alain April, Alfredo Benso, Laxmi Bhuyan, Greg Byrd, Robert Dupuis, David S. Ebert, Ming C. Lin, Linda I. Shafer, Forrest Shull, H.J. Siegel

MAGAZINE OPERATIONS COMMITTEE

Forrest Shull (chair), M. Brian Blake, Maria Ebling, Lieven Eeckhout, Miguel Encarnação, Nathan Ensmenger, Sumi Helal, San Murugesan, Yong Rui, Ahmad-Reza Sadeghi, Diomidis Spinellis, George K. Thiruvathukal, Mazin Yousif, Daniel Zeng

Editorial: All submissions are subject to editing for clarity, style, and space. Unless otherwise stated, bylined articles and departments, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *IEEE Software* does not necessarily constitute endorsement by IEEE or the IEEE Computer Society.

To Submit: Access the IEEE Computer Society's Web-based system, ScholarOne, at http://mc.manuscriptcentral.com/sw-cs. Be sure to select the right manuscript type when submitting. Articles must be original and not exceed 4,700 words including figures and tables, which count for 200 words each.

IEEE prohibits discrimination, harassment and bullying: For more information, visit www.ieee.org/web/aboutus/whatis/policies/p9-26.html.

SUSTAINABLE FORESTRY INITIATIVE Certified Sourcing
www.sfiprogram.org
SFI-01042