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

Extending Our Field's Reach

Diomidis Spinellis

I RECENTLY COLLABORATED with a digital-typography expert to create a formatting style for a publishing project. I thought we had agreed to work together through GitHub, which would let us share the current version of the manuscript and easily integrate the LaTeX style files by merging their corresponding development branch. Instead, a few weeks later, I received a .zip archive file over email.

> Almost all software engineering processes can benefit industries that work with executable knowledge.

The style was beautiful, with great attention to detail and typographical niceties. The archive's contents were also interesting. It contained two directories with similar but not identical contents: one for all projects requiring the specific style and one slightly tailored to my needs. Each directory contained tens of third-party files (some in binary format), log files, documentation, and automatically generated files. The style's source code files also contained things that might trouble a software engineer: outdated or inconsistently indented comments, copy-and-pasted and commented-out code, and a few overly long lines. To be fair, some of the style code seemed to have been written more than two decades ago, and we all know how software ages.

Still, whenever I collaborate with publishers, I'm always saddened to see the opportunities for process improvement they miss by not using the best practices and tools we've established in software engineering. For example, instead of collaboration using simple text markup over an online revision control and review system, documents in diverse incompatible binary formats are shuttled back and forth over email and, yes, FTP, with changes and comments embedded obscurely (sometimes with typographical marks scribbled on the margins of scanned paper). This creates an integration nightmare, which is only partially controlled by draconian, inflexible change-management policies and heroic efforts of all the involved parties. Once a document leaves a specific stage—say, drafting, copy-editing, or composition—there's no going back, and nobody can trace changes on an end-to-end basis. I know that some publishers use version control systems, but even there, due to the lack of build-process automation, such use often degenerates into that of a shared disk drive.

Picking on publishing would be wrong; other industries are also producing what's in effect software (ex-

 0740-7459/15/$31.00 © 2015 IEEE

ecutable knowledge) but not treating it as such. As examples, consider 3D printing, numerical control of machine tools, filmmaking, pharmaceutical laboratory automation, and workflow management. Other activities, with more abstract output, include project planning and drafting laws and regulations. The output of these activities shares considerable similarities with software: laws are constantly revised and refer to parts of each other, similarly to subroutines. These examples don't include devices in which software (the kind that runs on a CPU) is taking an ever-increasing role, such as cars, planes, phones, medical devices, and TVs. There, the problems and missed opportunities are much less severe; trained software engineers typically perform and manage the development.

Although many industries have developed their own highly effective processes over the years, software engineering maintains an essential advantage. It has developed methods and tools that let even small teams manage extremely high complexity. For example, compare the nine million LOC in the Linux kernel (which often forms only a small part of a much larger software stack) with the few tens of thousands of components in a modern car. This advantage is important because the complexity in nonsoftware activities is also increasing inexorably. Films used to have a few hundred scenes and takes, which filmmakers could easily manage by writing on a clapperboard. In contrast, modern blockbusters might depend on tens of thousands of digital artifacts. Also, processes that skilled humans executed only a few times in the past (for example, creating a car model for aerodynamic testing) can now be easily rerun (for example, by a 3D printer) with the touch of a button.

Almost all software engineering processes can benefit industries that work with executable knowledge. Requirements engineering can improve analysis, specification, validation, and traceability (why do we drill this hole at the bottom of the engine block?). Software design can be essential to control complexity through modeling, abstraction, control of coupling and cohesion, decomposition, encapsulation, and separation of concerns.

Returning to the example I presented at the beginning, the two directories I received from the publisher should have been divided into a meaningful hierarchy (decomposition) and their contents united through some application of polymorphism. Construction techniques can be used for the promotion of agility, build automation, continuous integration, reuse management, and verification. Imagine a film director being always able to use continuous integration to view the most current version of a film as it develops over the months. Software-testing techniques can reduce waste and increase efficiency, while software-inspired maintenance activities can increase a product's or process's longevity.

I believe that configuration management tools and techniques are the most productive way through which software engineering can affect other fields. The powerful tools and platforms we've developed (think of Git and GitHub) allow the effortless sharing of work over distance and time zones, the documentation and traceability of changes, the integration of issue and change management, commenting on each other's work, the organized development of separate product lines, and the pain-free merging of work items and branches. Engineers in other fields often look at these feats as alien technology. Some people, though, are recognizing the potential; consider the appearance of French civil code and German laws on GitHub (https://github.com/steeve/france.code-civil and https://github.com/bundestag/gesetze).

Exporting our hard-earned knowledge to other fields won't be easy. Each has distinct goals, competencies, values, and traditions. To communicate effectively, we must develop a shared vocabulary and way of thinking. Perhaps education is the easiest path: train practitioners from other disciplines to think and act as software engineers.

Software engineering has benefited mightily from research in fields ranging from electrical engineering and physics to mathematics and management science. It has changed our world beyond recognition by putting the artifacts it produces on billions of devices. Now, the time has come to transform our world in another way, by giving back to science and technology the knowledge software engineering has produced in the past half century.

HOW TO REACH US

WRITERS

For detailed information on submitting articles, write for our editorial guidelines (software@computer.org) or access www.computer.org/software/author.htm.

LETTERS TO THE EDITOR

Send letters to

Editor, *IEEE Software*
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
software@computer.org

Please provide an email address or daytime phone number with your letter.

ON THE WEB

www.computer.org/software

SUBSCRIBE

www.computer.org/software/subscribe

SUBSCRIPTION CHANGE OF ADDRESS

Send change-of-address requests for magazine subscriptions to address.change@ieee.org. Be sure to specify *IEEE Software.*

MEMBERSHIP CHANGE OF ADDRESS

Send change-of-address requests for IEEE and Computer Society membership to member.services@ieee.org.

MISSING OR DAMAGED COPIES

If you are missing an issue or you received a damaged copy, contact help@computer.org.

REPRINTS OF ARTICLES

For price information or to order reprints, email software@computer.org or fax +1 714 821 4010.

REPRINT PERMISSION

To obtain permission to reprint an article, contact the Intellectual Property Rights Office at copyrights@ieee.org.