Editor: **Philippe Kruchten**
University of British Columbia
pbk@ece.ubc.ca

Bringing the Human Factor to Software Engineering

Luiz Fernando Capretz

SOFTWARE IS A byproduct of human activities that incorporates our problem-solving capabilities, cognitive aspects, and social interaction. However, humans are more complicated and less predictable than software—and some of our complexity creates intricate dynamics in the software development process that cannot be ignored. Because of its multifaceted aspects, software development is among the most difficult tasks performed by humans today.

I'm not saying that technical skills are less relevant to a software project's successful outcome; rather, the human factor is a make-or-break issue that affects most software projects. I've witnessed software engineer "stars" unable to work together toward a common goal due to personality clashes, and I've seen average software engineers perform outstanding work because they enjoyed doing their tasks and gelled as a team. This indicates that software engineering boils down to technical competence and human factors. However, technical skills and soft skills don't receive the same degree of attention, especially by instructors of technical knowledge. It's important to understand why this occurs and what can be done to remedy the situation.

The cross-section of human and technical factors isn't new. Two pioneers (Gerald Weinberg and Ben Shneiderman) astonished many programmers in the 1970s by bringing topics to light, such as programming as human performance, in which they examined the facets of a good programmer. Specifically, they explored programming as a social and individual activity, defined egoless programming, and looked at the personality factors that impacted programming. Occasionally, papers present quantitative and qualitative research on related topics, but we've only scratched the surface on the impact of human factors in the software development process. It's often overlooked by educators and practitioners because human factors are usually related to soft skills, not rocket science or hardcore engineering.

Why Is This Happening?

Although the human factor is a topic that attracts interest from the general public, it's a new venture for many technical people. They might be afraid of making the wrong assumptions, not have time for this type of investigation, or undervalue the field, deeming it to be unimportant or irrelevant. Others might think that there's no place for the discussion of soft skills in an area that's so

continued on p. 102

0740-7459/14/$31.00 © 2014 IEEE

continued from p. 104

driven by logic and based on a mathematical foundation. These misconceptions discourage educators and researchers willing to work in this interdisciplinary area.

From management's perspective, an understanding of human factors is important in the context of the practice of software engineering. For example, managers need to learn about such topics because they frequently deal with negotiations and personality conflicts among team members, value the importance of hiring talented software engineers for the right job, and appreciate people able to function within a team. Nevertheless, for the majority of software practitioners, human factors are still considered marginal and treated as common sense.

That said, people sometimes struggle to remember that we're dealing with creatures of logic and emotions, not just ones and zeros. If we open our minds to research questions on gray areas, more researchers might be stimulated to run studies, validated through experimentation, to prove whether "gray claims" are right, wrong, myths, or half-truths; these studies can eventually converge to provide the best answer.

What Can We Do?

Software engineering is essentially a human activity, not just a technical matter of technology, and yet because of the emphasis placed on the technical aspects of software production, most software engineers have never considered software construction in this light. To evolve, we must examine our discipline through new lenses, from several perspectives. Software professionals should also delve into nontechnical issues and recognize that the people involved in the software development process are as important as the processes and the technology itself.

Team members should also recognize that software development is a sociotechnical practice. The software industry needs to encourage internal empirical experiments to understand human factor issues within performance-oriented teams, collecting data and creating insights to improve the overall development process. These studies could lead to improvements in products, increases in productivity, or decreases in production costs, thus indirectly affecting the company's bottom line. Whenever I meet senior software engineers who lead teams, they always hint at their need for that kind of soft skillset among software developers.

Although software engineering curricula vary greatly, there's now a common acceptance that they should be based on the guidelines for undergraduate degree programs in software engineering, a joint initiative of the IEEE CS and ACM Computing Curricula Software Engineering (http://sites.computer.org/ccse/SE2004Volume.pdf). A quick search for "human factors" in that guideline reveals that the term appears six times in the document body; a prominent sentence reinforces the importance of the topic: "Students need to repeatedly see how software engineering is not just about technology." But reality indicates that, at best, human factors are squeezed into a couple of courses, such as technical communication, software project management, or software process.

This won't change until we realize that the human element is pivotal to software engineering and that it's worthwhile studying and teaching this so-called soft subject. However, few courses in any computer science or software engineering curricula even mention it. An entire course on the human aspects of software engineering (which would be ideal) exists in very few universities, but a much easier path might be to include a few lecture hours about this topic in an existing undergraduate course, whether it's on programming, testing, design, or project management. I introduce the topic in my senior course on software verification and

validation in a lecture-discussion format; the feedback from students is phenomenal. I also borrow ideas from Carl Jung's theory on personality types when coordinating a capstone project course (Software Engineering Design II). This broadens students' horizons and ensures that they become more practice-ready when they enter the profession; ironically, students are able to fully understand the rationale behind these topics when they start working in the software industry.

A graduate course also offers the opportunity to teach these concepts in a seminar format that requires students to read about specific topics in advance and then engage in discussions. Graduate students are more mature and those with experience in the industry can easily relate their daily work experience to such topics. In my graduate course (Advanced Topics in Software Engineering), I go one step further and ask students to write short papers on related topics. A longer course could require them to run simple experiments with human subjects and report results.

More master's and PhD students can be stimulated to conduct research in this interdisciplinary area. Critical human-technical issues in software engineering can be investigated and students can be guided in designing, executing, and analyzing controlled empirical studies in a novel fashion. However, any serious scientific work would require some background in behavioral science experimentation, a strong foundation in empirical research methods, knowledge of statistical analysis, and approval from an ethics board to conduct pilot experiments in cases where sensitive information on human subjects is collected.

A holistic approach would involve getting some like-minded people together and starting tracks and special issues within well-known software engineering conferences and journals, encouraging researchers and software professionals to submit papers on human-centered topics. Concurrently, materials about mental processes, team interaction, appraisal and motivation, techno stress, personality, and sociotechnical issues could be inserted into undergraduate and graduate courses, and behavioral science topics could be slowly introduced under the umbrella of HCI and software management courses. Justifying the need for such topics shouldn't be too difficult—they would receive positive feedback from those involved in the industry.

As many software managers can attest, major failures in software projects eventually come down to people; in spite of this fact, the human aspects of software engineering don't receive the attention they deserve. Although the study of human factors in software engineering won't be a silver bullet that solves all problems, it will offer different insights and fresh approaches to answering many open questions in software engineering. Diversity of people and ideas are good for our field. Try it out!

LUIZ FERNANDO CAPRETZ is a professor of software engineering and assistant dean (IT & e-Learning) at Western University in Canada, where he also directed a fully accredited software engineering program. He has vast experience in the engineering of software and is a licensed professional engineer in Ontario. Contact him at lcapretz@uwo.ca or via www.eng.uwo.ca/people/lcapretz.

IEEE Software (ISSN 0740-7459) is published bimonthly by the IEEE Computer Society. IEEE headquarters: Three Park Ave., 17th Floor, New York, NY 10016-5997. IEEE Computer Society Publications Office: 10662 Los Vaqueros Cir., Los Alamitos, CA 90720; +1 714 821 8380; fax +1 714 821 4010. IEEE Computer Society headquarters: 2001 L St., Ste. 700, Washington, DC 20036. Subscription rates: IEEE Computer Society members get the lowest rate of US$59 per year, which includes printed issues plus online access to all issues published since 1984. Go to www.computer.org/subscribe to order and for more information on other subscription prices. Back issues: $20 for members, $216.17 for nonmembers (plus shipping and handling).

Postmaster: Send undelivered copies and address changes to *IEEE Software*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the USA.