Behavioral Science of Software Engineering

Marian Petre, The Open University

Jim Buckley, University of Limerick

Luke Church, University of Cambridge

Margaret-Anne Storey, University of Victoria

Thomas Zimmermann, Microsoft Research

Digital Object Identifier 10.1109/MS.2020.3014413
Date of current version: 22 October 2020

LARGE-SCALE SOFTWARE DEVELOPMENT is a sociotechnical activity only bounded by human imagination, ingenuity, and creativity. It involves teams of developers progressing by coordinating their activities and communicating their bottlenecks, goals, and advancements toward the wider goal of creating large, high-quality software systems. The stakeholders they serve are diverse (for example, clients, infrastructure providers, open source communities, project managers, and regulatory authorities), and often they have many competing, implicit requirements. But, as the political and legal implications of algorithms and data (https://harvardmagazine.com/2000/01/code-is-law-html) increasingly affect society, it is imperative that the systems the developers build are high quality in terms of accurately embodying all of those requirements.

Consequently, understanding human reasoning and the social context in the software engineering process is crucial, promoting innovation, productivity, and quality. There is a well-established, international community that conducts empirical studies of the psychology of software engineering, applying cognitive and social psychological theory to software development to make sense of practice, and to lead to new insights, methods, and tools.

Researchers in both industry and academia have been studying the cognitive, social, and behavioral aspects of software development for at least 50 years. As outlined by Blackwell et al.,[1] early work in the 1970s focused on the cognitive work done by programmers. For example, Weinberg's *The Psychology of Computer Programming* was first published in 1971,[2] and the first paper to directly address the psychology of programming in the *International Journal of Man Machine Studies* (subsequently the *International Journal of Human–Computer Studies*) was "Psychological Evaluation of Two Conditional Constructions Used in Computer Languages" by Sime et al.[3] Contemporaneously, the relevance of the human and behavioral aspects of software development was highlighted by Brooks' classic *The Mythical Man-Month*.[4]

This trend of a cognitive approach continued through the 1980s, with a much-expanded range of studies applying psychological methods to the study of software development at scale. By the 1990s, the focus had shifted from individual cognition to situated practice, drawing on social psychology to address professional skills and contexts. The focus expanded in the early 2000s to the social enterprise of software development, drawing on the range of behavioral sciences to study bigger developments by larger development teams, including distributed teams in a global context.

This perspective has been explored extensively within a range of communities including, but by no means limited to the

- IEEE/ACM International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)
- ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)
- PLATEAU workshop (evaluation and usability of programming languages and tools)
- ACM Conference on Computer-Supported Collaborative Work and Social Computing (CSCW)
- ACM Conference on Human Factors in Computer Systems (CHI)
- Psychology of Programming Interest Group (PPIG)
- IEEE Symposium on Visual Languages and Human-Centered Computing (VL/HCC).

However, within many software engineering venues, such as the Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) and the International Conference on Software Engineering (ICSE), research remains primarily focused on the technical aspects of the tools and processes, without considering humans in their evaluations.[5] That is not to say that all studies should be exclusively human centric, but rather that "…there is a need for strategies that aim at a deeper understanding of human and social aspects of software development practice to balance the design and evaluation of technical innovations."[5] The raft of behavioral science approaches available to researchers means that this need can be addressed toward a more holistic understanding of software development and thus identification of opportunities for improvement in the process and the product.

This special issue of *IEEE Software* aims to provide a snapshot of how these worlds and approaches can meet. As illustrated in Table 1, the topics discussed are widely diverse, tackling a range of important software engineering challenges. These include the improved incorporation of requirements into Agile software development (Sedano, Ralph, and Péraire) and addressing the information needs of developers tasked with software evolution in general (LaToza). Other topics include addressing resistance in software projects (Cheikh-Ammar, Bourdeau, and

Table 1. The articles in this issue use a range of methods to address diverse software engineering challenges from a behavioral science perspective.

Authors	Title	Topic	Methods
Baltes, Park, and Serebrenik	"Is 40 the New 60? How Popular Media Portrays the Employability of Older Software Developers"	Perceptions of age and employability	Analysis of online media
Nagaria and Hall	"Reducing Software Developer Human Errors by Improving Situation Awareness"	Situation awareness and error reduction	Training experiment
Cheikh-Ammar, Bourdeau, and Darveau	"Navigating the Rough Seas of Software Project Resistance"	Strategies for project management within software teams	Semistructured interviews
Jaspan et al.	"Enabling the Study of Software Development Behavior With Cross-Tool Logs"	Software development productivity	Behavioral logging
LaToza	"Information Needs: Lessons for Programming Tools"	Cognitive and information needs of developers	Analysis of tools
Sedano, Ralph, and Péraire	"Dual-Track Development"	Human-centric design in Agile software development	In situ, industry-based evaluation
Guizani et al. ("Requirements" department)	"Gender Inclusivity as a Quality Requirement: Practices and Pitfalls"	Gender inclusivity in software	In situ, industry-based evaluation

Darveau) and reducing errors in software by providing developers with relevant contextual information and illustrating how logging developer behavior can lead to increased developer productivity and team building (Jaspan et al.). They also cover topical, contextual issues, such as ageism (Baltes, Park, and Serebrenik) and gender inclusivity (the "Requirements" department) in software engineering, toward providing a more balanced inclusive makeup of development teams.

Likewise, these articles illustrate the range of behavioral science approaches available to researchers, from in situ, industry-based studies (such as the article by Sedano, Ralph, and Péraire as well as that by Guizani et al.) to more formal experiments (Nagaria and Hall). Between these two extremes are reviews of tools (LaToza), analysis of social media (Baltes, Park, and Serebrenik), interviews (Cheikh-Ammar, Bourdeau, and Darveau), and a Google study of the ethical behavioral logging of software developers' activities (Jaspan et al.).

In summary, although the diversity of software engineering topics addressed shows the applicability and importance of the behavioral science lens to the study of software engineering, the diversity of behavioral science approaches employed by the authors illustrates the breadth of approaches, methods, and analytic tools available to researchers when they try to develop a deeper understanding of software engineering concerns.

The articles are summarized as follows. Baltes, Park, and Serebrenik analyzed the public discourse within the U.S. developer media about perceptions of age and employability. They looked at both relevant online articles, and the discussions about them in *Hacker News*, and found that many developers are now considered "old" at 40+ years of age. They identified both the perceived employment issues and the strategies used to mitigate them, both strategies associated with technical skills (such as specialization and mastering modern technologies) and those associated with social perceptions of the individual or the company culture.

Nagaria and Hall studied the potential of *situation awareness* training to reduce errors during software development by enabling the retention of contextual knowledge during task performance. They developed an online situation awareness training package based on the cycle observe–orient–decide–act (OODA) loop.[6] They evaluated the effect in a preliminary experiment with professional developers that tested their in situ development behavior over five days before and five days after training. Their preliminary results

ABOUT THE AUTHORS

MARIAN PETRE is a professor of computing at the School of Computing and Communications, The Open University. Contact her at m.petre@open.ac.uk.

MARGARET-ANNE STOREY is a professor of computer science and a codirector of the Matrix Institute for Applied Data Science, University of Victoria, Canada. Contact her at mstorey @uvic.ca.

JIM BUCKLEY is a senior lecturer at Lero/The Department of Computer Science and Information Systems at the University of Limerick, Ireland. Contact him at jim.buckley@ul.ie.

THOMAS ZIMMERMANN is a senior principal researcher, Microsoft Research, Redmond, Washington. Contact him at tzimmer@microsoft.com.

LUKE CHURCH is an affiliated lecturer, the Department of Computer Science and Technology, University of Cambridge, United Kingdom. Contact him at luke@church.name.

suggested that developer errors were reduced with OODA loop use.

Cheikh-Ammar, Bourdeau, and Darveau used interviews with experienced project managers to characterize the nature of resistance that occurs within software projects. They described how project managers can intervene by adopting one of four archetypal personas—the coach, doctor, politician, and priest—each of which supports different preventative and curative interventions to overcome resistance.

Jaspan et al. looked at behavioral logging of software developers and how that might inform software development practices. Specifically, they described a behavioral logging system developed at Google and discussed how that system can inform on topics as diverse as the "benefits of code-conventions training with respect to code reviews" and "identifying negative interpersonal interactions" in their teams.

LaToza focused more on cognitive aspects of individual developers, describing the information needs experienced by individual software engineers when onboarding open source projects, when navigating and debugging code, and when trying to uncover aspects of the system's design rationale. He pointed to several tooling initiatives that address these information needs congruently, as illustrations of the impact understanding human reasoning can have in software development.

Sedano, Ralph, and Péraire reviewed focused how the software development process can be reconfigured to allow greater incorporation of human-centered design in projects that follow an Agile method. Their approach has been refined/evaluated in situ over three years in a commercial organization (Pivotal), and it has been well received by developers who attribute the success of projects to the "constellation of practices" described.

In the "Requirements" department, Guizani et al. reported on the use of their tool GenderMag (a method for detecting and fixing gender inclusivity issues in software) by 10 professional software development teams. Drawing on longitudinal data collection, their column summarized key practices and pitfalls observed in use.

These articles represent just a fraction of the more than 50 submissions to the special issue, highlighting again the broad span of excellent research occurring in this space. This is further highlighted by the "Practitioners Digest" department, which details increased activity in fora such as ICSE, for example. Unfortunately, we could not include all the excellent submissions we received in this special issue, but we have recommended them for inclusion in future issues of *IEEE Software*, to further illustrate the possibilities for applying behavioral methods to understanding and supporting software development practice, across different developers, contexts, and research questions.

However, some important discussions were missing. The study of social and behavioral aspects of developing with new emerging technologies, such as machine learning, was largely absent from the submissions, as were discussions on the emerging politics of algorithms and the negotiations of the role of software in society beyond legal and regulatory frameworks. As the work on inclusiveness and diversity shows, these issues are now at the forefront of modern software practice. We look forward to future research growth in these important spaces, with progress reflected in future submissions to *IEEE Software*. SW

References

1. A. F. Blackwell, M. Petre, and L. Church, "Fifty years of the psychology of programming," *Int. J. Human–Comput. Stud.*, vol. 131, pp. 52–63, Nov. 2019. doi: 10.1016/j.ijhcs.2019.06.009.
2. G. M. Weinberg, *The Psychology of Computer Programming*. Dorset House, 1971.
3. M. E. Sime, T. R. G. Green, and D. J. Guest, "Psychological evaluation of two conditional constructions used in computer languages," *Int. J. Man Mach. Stud.*, vol. 5, no. 1, pp. 105–113, 1973. doi: 10.1016/S0020-7373(73)80011-2.
4. F. Brooks, *The Mythical Man-Month*. Reading, MA: Addison-Wesley, 1975.
5. M. Storey, N. A. Ernst, C. Williams, and E. Kalliamvaku, "The who, what, how of software engineering research: A socio-technical framework," *Empirical Softw. Eng.*, vol. 25, no. 5, pp. 4097–4129. doi: 10.1007/s10664-020-09858-z.
6. J. Boyd, "A discourse on winning and losing [Briefing Slides]," Air Univ. Library. Maxwell Air Force Base, AL, Document No. MU 43947, 1987.

Digital Object Identifier 10.1109/MS.2020.3027396