Editor in Chief: **Ipek Ozkaya**
Carnegie Mellon Software Engineering Institute,
ipek.ozkaya@computer.org

# The Behavioral Science of Software Engineering and Human–Machine Teaming

Ipek Ozkaya

**DESIGNING AND SUSTAINING** sociotechnical systems where relationships among humans, machines, and environmental aspects are intertwined is not new to software engineering. Emery and Trist[1] coined the term *sociotechnical systems* in 1960 to draw attention to the need for people, machines, and context to all be considered when developing and sustaining these systems. Interactions and dependencies in sociotechnical systems get complex quickly as the interdisciplinary nature of such systems drive different design priorities and information flow mechanisms: sociologists see social systems, psychologists observe them as cognitive systems, computer scientists approach them as information systems, and engineers see the hardware systems.[2] All of these perspectives are not only valid but also are essential elements of sociotechnical systems.

The behavioral science of software engineering focuses on the cognitive, social, and behavioral implications of developing software systems.[3] In a recent publication, Storey and colleagues[4] examined 151 software engineering papers published in two premium software engineering venues, the International Conference on Software Engineering and *Empirical Software Engineering Journal*. They observed that, while the findings cited in the papers claimed to focus on people as part of their research, they often did not include explicit consideration of human aspects. These findings demonstrate that while software engineers recognize that software systems are part of the sociotechnical systems in which humans and their behavior are part of the system design, we still lack a clear emphasis on incorporating the study of humans into the process of design.

> The behavioral science of software engineering focuses on the cognitive, social, and behavioral implications of developing software systems.

The sociotechnical systems of the future without doubt will also include

*Digital Object Identifier 10.1109/MS.2020.3019190*
*Date of current version: 22 October 2020*

## CONTACT US

**AUTHORS**

For detailed information on submitting articles, visit the "Write for Us" section at www.computer.org/software

**LETTERS TO THE EDITOR**

Send letters to software@computer.org

**ON THE WEB**

www.computer.org/software

**SUBSCRIBE**

www.computer.org/subscribe

**SUBSCRIPTION CHANGE OF ADDRESS**

address.change@ieee.org
(please specify *IEEE Software*.)

**MEMBERSHIP CHANGE OF ADDRESS**

member.services@ieee.org

**MISSING OR DAMAGED COPIES**

help@computer.org

**REPRINT PERMISSION**

IEEE utilizes Rightslink for permissions requests. For more information, visit www.ieee.org/publications/rights/rights-link.html


artificial intelligence (AI) components. Smith[5] emphasizes that designing trustworthy AI systems and human–machine teaming has to start from an explicit and consciously designed inclusion of human aspects. Understanding human reasoning and cognition has always been crucial in software engineering to better design for the complex interactions between users and systems. However, we are entering a new era where the behavioral science of software and system engineering must increasingly both guide design principles of sociotechnical systems and focus explicitly on human–machine teaming. How technologies will interface with humans to establish effective human–machine teaming requires an understanding of how various engineers, developers, and end users behave as well as an understanding of the uncertainty involved in the behavior of AI-enabled systems.

Designing trustworthy AI systems and human–machine teaming has to start from an explicit and consciously designed inclusion of human aspects.

## What Is Human–Machine Teaming?

The term *human–machine teaming* refers to the efficient and effective integration of humans with complex machines. While it is easy to assume that any user interaction with any user system is human–machine teaming, our emphasis should be on teaming rather than just human–machine interactions. Effective teaming implies having a shared awareness of the task, team, and context as well as some shared commonality and understanding of the end goal to be achieved. In a recent report, 605 U.S. workers were asked to identify an intelligent technology they use on a regular basis and classify the interaction with that technology as a teammate or a tool. In this study, 68% of the respondents classified the intelligent technologies they employed, ranging from autonomous cars, service robots, industrial robots, robotic assistants, and navigation aids to small home intelligent devices, as tools rather than teammates. The lack of decision authority and communication richness was among the top reasons why participants viewed the technology as a tool instead of a teammate.[6]

## Changing Interactions and the Mental Model of Users

How user interaction models will need to evolve when considering human–machine teaming is currently insufficiently studied in behavioral science of software engineering. A top priority concern in designing effective human–machine teaming is trust: whether humans will and should trust the systems to make decisions on their behalf or collaboratively.

The interaction models of humans with computers will and should change. Improving our understanding of what effective and trustworthy human–machine teaming looks like will shape the design of interactions. Researchers will also need to better understand how human–machine interactions will deviate from current design models and consequently develop new models.

Software systems influence human cognition and task flows; how those task flows should be modified is not always predetermined or even understood despite all the contextual design focus when constructing systems.[7] A software system as a tool creates new task flows. The ultimate goal of any software system is to improve the effectiveness of its users in completing their tasks. Successful systems are those that augment human behavior in more efficient ways or sometimes define a completely new way for people to achieve their tasks. An example of this phenomenon was observed when CAD tools became available to engineers and designers after their first introduction as a concept with Sketchpad in 1963 by Ivan Sutherland.[8] CAD tools work with the mental model of repetition, reuse, and scaling to the rest of the system that is being designed.

CAD broke the barrier between the act of designing and that of creating the blueprint artifact. However, the engineers and designers who are the target users for these tools had to learn to approach their task differently. They needed to recognize the reused elements of their designs so that they could create once and propagate as needed. CAD tools influenced design capture and, in a way, eliminated barriers, allowing quicker iterations and approaching design as an activity where repetitive elements are proactively recognized.[9]

CAD tools enabled new interaction flows to be accepted by end users by focusing on their goals: to iterate on designs at ease and create the artifact along the way without an added burden. Consequently, as the users became familiar with these tools, they were able to allocate more time to the design activity. We will likely observe similar task shifts as we gain more experiences in human–machine teaming through the development and use of AI-enabled systems, in particular autonomous systems. For example, how quickly should a human react to a potentially wrong recommendation from the system, which recommendations are more essential to react to, how should users redirect their attention, and how can systems be designed to best support their human counterparts for effective teaming? We are yet to understand the limitations and horizons of humans in this new mode of human–machine teaming.

## The Behavioral Science of Software Engineering: Implications for Human–Machine Teaming

There are a number of implications for those studying the behavioral science of software engineering as well as those developing systems that will need to incorporate human–machine teaming. Software developers, software engineering researchers, data scientists, and engineers will need to do the following:

- Consider human aspects explicitly, with a focus on how their task flows may evolve and



*Digital Object Identifier 10.1109/MS.2020.2972718*

Defining the boundaries of trust when human–machine teaming is involved will improve the capabilities of the systems developed.

whether such changes are acceptable and within the goals of the outcomes expected from the systems.

- Start with a clear understanding of trust within the context of the system, from the perspective of end users, and design to that level of verifiable trust. Humans have different tolerance levels of trust depending on the system they are using. Defining the boundaries of trust when human–machine teaming is involved will improve the capabilities of the systems developed.
- Recognize that the systems developed may imply new interaction models where people may need to be retrained or the systems may need to be redesigned to improve the task flows for most effective human–machine interaction.
- Recognize that human–machine teaming goes beyond human–machine interaction and expectations such as trust, ethics, privacy, and control not only take priority as part of the behavioral science of software engineering but should also drive the system design.

Smith[10] shares an initial human–machine teaming framework checklist and agreement for teams who are designing for human–machine teaming. These can serve as a good starting point for both software engineers and behavioral scientists.

## References


1. F. E. Emery, E. L. Trist, C. W. Churchman, and M. Verhulst, "Socio-technical systems," in *Management Science: Models and Techniques*, vol. 2, C. W. Churchman and M. Verhulst, Eds. Oxford, UK: Pergamon, 1960, pp. 83–97.
2. G. D. Baxter and I. Sommerville, "Socio-technical systems: From design methods to systems engineering," *Interact. Comput.*, vol. 23, no. 1, pp. 4–17, 2011. doi: 10.1016/j.intcom.2010.07.003.
3. M. Petre, J. Buckley, L. Church, M.-A. Storey, and T. Zimmermann, "Behavioral science of software engineering," *Software*, vol. 37, no. 6, pp. 21–25. doi: 10.1109/MS.2020.3014413.
4. M. Storey, N. A. Ernst, C. Williams, and E. Kalliamvaku, "The who, what, how of software engineering research: A socio-technical framework," *Empir. Softw. Eng.*, vol. 25, no. 5, pp. 4097–4129, 2020. doi: 10.1007/s10664-020-09858-z.
5. C. J. Smith, "Designing trustworthy AI: A human-machine teaming framework to guide development," 2019. [Online]. Available: http://arXiv:CoRRabs/1910.03515
6. J. Lyons, K. Wynne, S. Mahoney, and M. Roebke, "Trust and human-machine teaming: A qualitative study," in *Artificial Intelligence for the Internet of Everything*. Amsterdam, The Netherlands: Elsevier, 2019, pp. 101–116.
7. K. Holtzblatt and H. Beyer, *Contextual Design: Defining Customer-Centered Systems* (Interactive Technologies). San Mateo, CA: Morgan Kaufmann, 1997.
8. I. Sutherland, "Sketchpad: A man-machine graphical communication system," Univ. Cambridge, Tech. Rep. UCAM-CL-TR-574, Sept. w2003.
9. S. K. Bhavnani and B. E. John, "Exploring the unrealized potential of computer-aided drafting," in *Proc. SIGCHI Conf. Human Factors Computing. Syst. (CHI'96)*, 1996, pp. 332–339. doi: 10.1145/238386.238538.
10. C. J. Smith, "Designing ethical AI Experiences: Checklist and agreement," Carnegie Mellon University, Software Engineering Institute, Pittsburgh, fact sheet, Dec. 2019. Accessed: Sept. 2020. [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=636620