

User Interface Design in the 21st Century

Antti Oulasvirta, Aalto University
Gregory D. Abowd, Georgia Tech

What should 21st-century UI-design principles, tools, and methods look like? The articles in this special issue present a variety of responses to this question, covering such areas as "Cuddly UIs," home robots, smart services, gesture recognition, and programming tools.

This special issue was sparked by the observation that user interface (UI) design might be becoming computing's weak spot. Consider some of the most heralded visions of computing, such as wearables, robots, machine intelligence, or the Internet of Things. Realizing them relies on solving design problems touching some very hard sociotechnical issues. How do we design an interface for a device or service that is placed on the user's skin, embedded in physical objects, or intertwined with social networks? How can we control intelligent embodied agents such as self-driving cars or household robots? How can people make sense of massive datasets? How can we help them understand and manage privacy in the era of ubiquitous computing, in which everything is networked and uses sensors?

Our ability to solve these emerging challenges relies on the principles, methods, and tools we use in UI design.

THE GUI-ERA PARADIGM

The contemporary UI paradigm was developed in the era of the PC and

its GUI, which emerged in the 1980s. The concept of direct manipulation exposed GUIs' defining feature: they decrease the perceptual and cognitive distance between a visually presented object and its computational correspondent.¹ This concept was the key to a number of novel visualizations and interaction techniques. Empirical research produced hundreds of results that were consolidated into heuristics for GUI design. Researchers translated these heuristics into checklist-like rapid-inspection methods developers could use to evaluate designs without expensive empirical studies. Usability engineering repackaged the whole user-centered design cycle for software engineering's needs.² Its centerpiece was usability testing—a streamlined version of experimental methods used in psychology. In parallel, researchers developed sketching methods and tools to boost the quality and rate of design ideation.³

These breakthroughs remarkably increased the quality of design outcomes and the reliability of design as a process. As IT companies started discovering usability's benefits and competing on the basis of usability, whole professions were rapidly born, including UI designers, usability evaluators, and interaction designers. Although current designers differentiate their profession by using the terms "user experience design" or "interaction design" instead of "UI design," they still largely focus on UIs and use GUI-era methods and concepts.⁴ They just use different terms.⁴

However, the depth of problems designers face has expanded considerably. Information systems design's traditional concerns—usability, usefulness, and satisfaction—are still

relevant. However, designers also face issues such as materials, physiology, values, artificial intelligence, big data, robots, and sociotechnical systems, to name a few. It is unclear whether the GUI-era paradigm suffices to solve the new problems. We are justified to ask what 21st-century UI-design principles, tools, and methods should look like.

IN THIS ISSUE

This special issue received 16 submissions about UI design on the frontiers of computing. The selected five articles investigate topics with significant potential or risk, showing the maturity of new concepts and evidence of usefulness. Each article not only summarizes interesting research but also engages readers and directs future efforts.

In "Cuddly User Interfaces," Yuta Sugiura, Takeo Igarashi, and Masahiko Inami introduce a world in which computing is embedded in soft objects such as pillows and carpets. Such interfaces are a step toward ubiquitous computing. The authors challenge the regular rigid interfaces such as laptops and control panels and suggest how to integrate sensing and display in soft objects to make them interactive. They present four examples of using different computing methods to achieve the goal of finding a harmonious relationship between the computing devices and soft material.

In "Graphical Instruction for Home Robots," Daisuke Sakamoto, Yuta Sugiura, Masahiko Inami, and Takeo Igarashi investigate the control of robots as 21st-century appliances. Typically, human-robot interaction research has explored GUIs for robot control in such diverse domains as military missions and undersea operations. But that rich body of research does not trivially translate to interfaces




for nonhumanoid home robots. The authors describe three interaction designs for home robots. They conclude that graphical instruction will continue to be a default UI that accepts indirect, asynchronous operations. The graphical representation of appliances and instruction will help compensate for our lack of understanding of what intelligent appliances “think.”

In the article “In Search of Coproduction: Smart Services as Reciprocal Activities,” John M. Carroll, Jiawei Chen, Chien Wen (Tina) Yuan, and Benjamin V. Hanrahan present a human-oriented perspective on smart-service design. They argue that services can be smart by being reciprocal, enabling service recipients to actively participate in service production. In this view, human initiative and cooperation are resources for building smart services.

In “Programming with Examples to Develop Data-Intensive User Interfaces,” Jun Kato, Takeo Igarashi, and Masataka Goto investigate programming for applications such as robot control, gesture recognition, image processing, and animation. Programming with Examples is a workflow that integrates graphical representations into integrated development environments (IDEs) to allow visualization and interactive editing of example data. Whereas other IDEs are usually only for programmers, this approach can bring nonprogrammers into the development process because people often can comprehend graphics even if they have no programming knowledge.

Finally, in “Programmers Are Users Too: Human-Centered Methods for Improving Programming Tools,” Brad A. Myers, Andrew J. Ko, Thomas D. LaToza, and YoungSeok Yoon address

how to use human-centered approaches during the requirements analysis, design, development, and evaluation of programming tools, to increase the tools’ usability and effectiveness. They discuss 10 human-computer interaction methods and principles, along with five usability recommendations. The results are based on years of research by Carnegie Mellon University’s Natural Programming group.

We hope these articles will spur readers to engage in further thought about the importance of human-centered design challenges that should proceed alongside advances in computing. 

ACKNOWLEDGMENTS

We thank David McGookin for assistance with reviewer recruitment.

REFERENCES

1. E.L. Hutchins, J.D. Hollan, and D.A. Norman, “Direct Manipulation

- Interfaces,” *Human-Computer Interaction*, vol. 1, no. 4, 1985, pp. 311-338.
2. J. Nielsen, *Usability Engineering*, Elsevier, 1994.
3. B. Buxton, *Sketching User Experiences: Getting the Design Right and the Right Design*, Morgan Kaufmann, 2010.
4. G. Lindgaard, “The Usefulness of Traditional Usability Evaluation Methods,” *Interactions*, vol. 21, no. 6, 2014, pp. 80-82.

ABOUT THE AUTHORS

ANTTI OULASVIRTA is an associate professor at Aalto University, where he leads the User Interfaces research group. His research focuses on computational approaches to user interface design. Oulasvirta received a PhD in cognitive science from the University of Helsinki. He is the editor of *Computer’s* Indistinguishable from Magic column. Contact him at antti.oulasvirta@aalto.fi.

GREGORY D. ABOWD is a Regent’s Professor and the J.Z. Liang Chair in Georgia Tech’s School of Interactive Computing. His research interests concern how the advanced information technologies of ubiquitous computing impact our everyday lives when they are seamlessly integrated into our living spaces. Abowd received a DPhil in computation from Oxford University. Contact him at abowd@cc.gatech.edu.



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