DOI:10.1145/3210753

In addition to having a detailed understanding of the artifacts they intend to create, designers need to guide the software tools they use.

BY STEFAN SEIDEL, NICHOLAS BERENTE, ARON LINDBERG, KALLE LYYTINEN, AND JEFFREY V. NICKERSON

Autonomous Tools and Design: A Triple-Loop Approach to Human-Machine Learning

» key insights

- **Autonomous tools are able to generate remarkable design outcomes, but designers using them also need to change the way they do their design work.**
- **Designing with autonomous tools requires that designers understand and actively interact with the "mental models" of the tools, in addition to the design artifact and the design process, what we call the "triple loop" model of learning.**
- **Designers working with autonomous tools need to build capabilities described here in terms of framing, evaluating, and adjusting to navigate this new design process.**

DESIGNERS INCREASINGLY LEVERAGE autonomous software tools that make decisions independent of the designer. Examples abound in virtually every design field. For example, semiconductor chip designers use tools that make decisions about placement and logic checking. Game designers rely on software that generates initial drafts of virtual worlds. Autonomous tools employ artificial intelligence methods, including machine learning, pattern recognition, meta-heuristics,

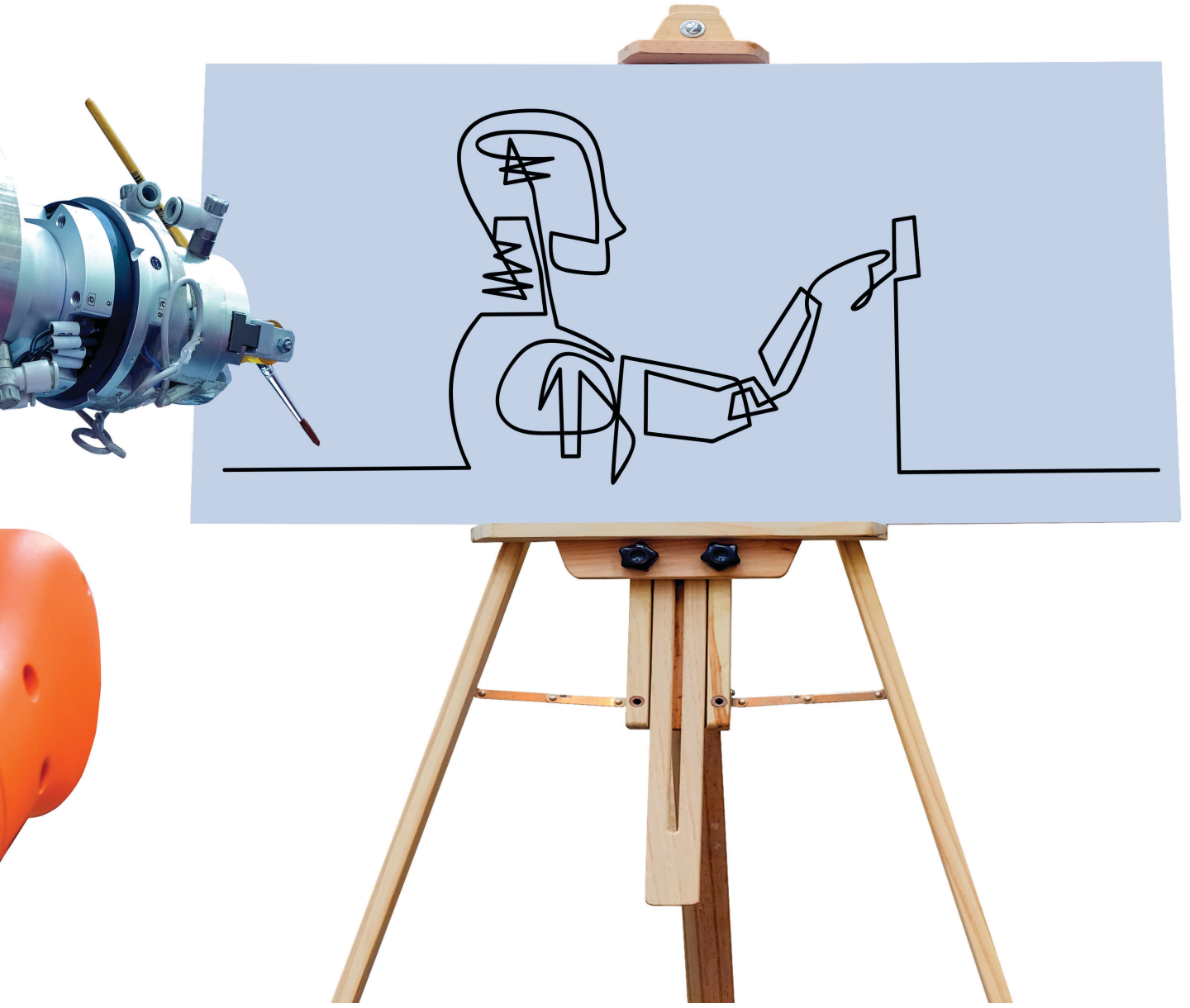


IMAGE COLLAGE BY ANDRIJ BORYS ASSOCIATES. USING SHUTTERSTOCK

and evolutionary algorithms to generate design artifacts beyond any human's capabilities.

A naïve view suggests these tools will someday replace human designers in the design process. An alternative perspective is that humans will continue to play an important role but also that this role is changing. To account for the changing role of humans in design processes powered by autonomous tools, we describe in this article a "triple-loop approach" to human-machine learning. Increasing amounts of design activity are most certainly being carried out by autonomous tools, but humans must still actively frame, evaluate, and adjust the "mental" models embedded in autonomous tools, in the form of algorithms.[a] Organizations employing autonomous tools in their design processes must thus account for these activities in their design processes.

a We say "mental model embedded in an autonomous tool" to indicate that just as human designers have mental models that guide their design activity, including their use of tools, autonomous tools also have models that guide their design activity. Both types of model change over time.

In what follows, we describe our triple-loop approach, followed by illustrative examples from research into the design of semiconductors, video games, software interfaces, and artificial intelligence. We conclude by identifying practices that enable designers to frame, evaluate, and adjust the mental models embedded in autonomous tools.

Design as Triple-Loop Human-Machine Learning

Design processes are a form of knowledge-intensive work that relies on designers' capacity to learn. In his semi-

nal work, Chris Argyris[2,3,4] explained how humans, in knowledge-intensive work, follow a double-loop process of learning. In the context of design work, the first loop involves learning about the design artifact. As designers experiment with alternatives, they correct errors and respond to feedback on design results (see Figure 1, loop 1). The second loop involves a designer's reflection on the ongoing process of design. Over time, designers learn, through reflection, to adjust their approaches and discover new processes and perhaps incorporate new tools that help them expand their thinking around the process of design. The second loop captures meta-level learning about the design process (see Figure 1, loop 2), highlighting how designers reflect on the mental models—goals, cognitive rules, and reasoning—they apply.

Triple-loop human-machine learning occurs whenever humans and autonomous computational tools interact in generating design outcomes. It is important for designers to understand how their own mental models interact with mental models embedded in the logic of autonomous tools. This process is distinct from conventional design work where tools are limited to supporting ongoing design tasks but do not play an independent role in shaping design outcomes.

Argyris calls mental models "master programs." In the case of designing with autonomous tools, the master program of the designer—the "designer's mental model"—may not be aligned with the master program of the autonomous tool, or "autonomous tool mental model," for a variety of reasons, including, for example, that the design activity usually involves more than one person; the designer using the tool is probably not the same person who programmed the tool; multiple designers may have different conceptions about what a master program does; and these conceptions may differ from what the programmers intended. Moreover, programmers may move on to other projects, along with the designers who originally informed the design of the tools; increasingly, neither the tool programmers nor the designers understand the implications of their decisions on what the tool is able to do. The mental models of designers and those embedded in autonomous tools as master programs for design activity thus capture a mutual learning process, suggesting a third loop in the classic model of the design process (see Figure 2).

It is important for designers to understand how their own mental models interact with mental models embedded in the logic of autonomous tools.

The first loop in the triple-loop model is similar to the original loop in the double-loop model, involving designers and tools interacting to generate design outcomes. However, in the triple-loop model, it is the tool that primarily generates the design alternatives. A given configuration of the tool generates alternatives from a set of input parameters and then evaluates them according to a set of predefined criteria.

The second loop can take two alternative forms—human learning or machine learning—as in Figure 2, loop 2a and loop 2b. From a human perspective, the second loop involves the human designer evaluating the alternatives and modifying input parameters, tool settings, and evaluation criteria for a given design problem in order to run the next round of generating design alternatives. The second loop, from a machine perspective, involves the program learning from designer feedback in the design process in order to modify itself and improve its model so it can generate better alternatives in subsequent rounds of design activity.

The third loop involves human designers learning about the mental model embedded in the tool and/or the tool learning about the human designers' mental models—through either explicit feedback or analyzing the usage patterns of the human designers. The machine models of designers are sometimes called "user models."[1] When machines run learning algorithms, the human designers may not fully understand the computations. What they thought the tool would do may or may not be what it actually does or was even designed to do, though designers collect feedback that can help them align their mental models and the mental models embedded in the tool (such as by adjusting the algorithm used). This process of learning about the mental model embedded in the autonomous tool and modification of the tool constitutes a second meta-level of learning during design processes involving

autonomous tools. Moreover, the tool may change its own model as it relates to what the human wants and how the human perceives the tool; this may result in changes in the interface or the design parameters being applied. Much like two people with different mental models learn from each other and work together to reconcile their mental models, autonomous tools and humans likewise have different models related to design goals and processes they may seek to reconcile through various loops of learning.

Illustrations

Here, we provide four examples of triple-loop human-machine learning, including in semiconductor design, software interface design, video game design, and artificial intellgence design. They highlight different aspects of the interaction between designers and autonomous tools.

Semiconductor design at Intel and AMD. Since the early 2000s, a new breed of tooling strategies based on genetic algorithms has emerged in semiconductor design,[6] commonly called "physical synthesis." Such tools offer a powerful way to improve the productivity of physical chip design by autonomously generating full layout solutions for whole sections of a chip (such as caches and USB ports). Major semiconductor manufacturers, including Intel and AMD, use the program-synthesis approach to generate full-layout designs of particular chip sections for a given set of parameter values. A program-synthesis designer starts each design cycle by defining a new set of design-parameter values that specify directions and constraints for the design search to be carried out by the tool (see Figure 3). When a solution is found through such search, the tool autonomously delivers a complete design solution for the given layout problem. After each such cycle, the designer manipulates the design by modifying the parameters based on the design outcomes generated during the previous cycle. Designers refer to the automated generation of design alternatives as "experiments" for a given set of parameters and interact with the algorithmic results in order to evaluate alternatives, given the input parameters and design goals (see Figure 2, loop 1). Designers then learn from the experiments in a way that helps them improve the input parameters for the next round of experiments, as in Figure 2, loop 2. Developers of the algorithms interact with the chip designers in order to learn how the chip designers change the parameters interactively after evaluating design outcomes. The developers learn about the effects of the mental models embedded in the tools, as well as the designers' mental models. This involves learning about the specific assumptions of the design-

Figure 1. Double-loop learning; based on Argyris.[3,4]

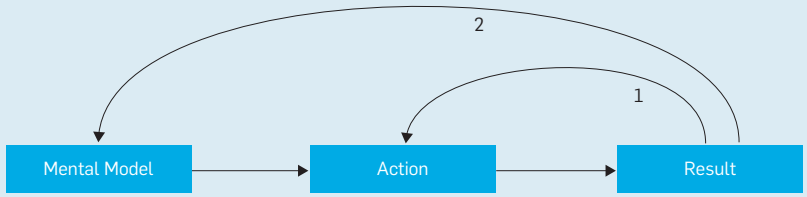


Figure 2. Triple-loop human-machine learning with autonomous tools.

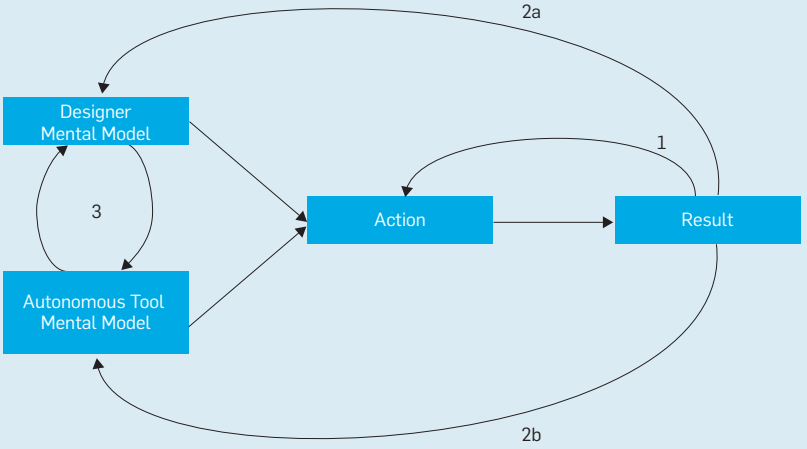
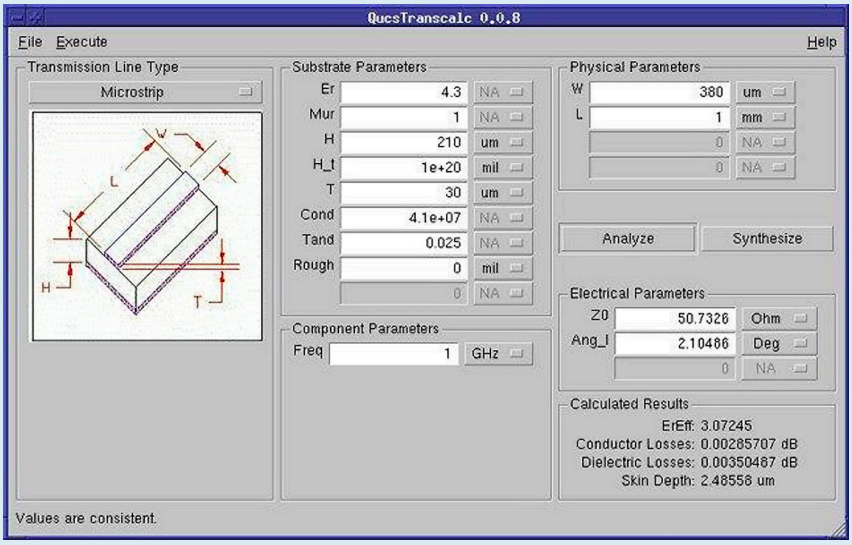


Figure 3. Computational design tool for semiconductor design.

"Quite Universal Circuit Simulator" is hosted on Sourceforge (http://sourceforge.net/projects/qucs/) and made available under the GNU General Public License version 2.0.



ers while rooting out inefficiencies of the tools by updating and rewriting the source code for the tools, as in Figure 2, loop 3. Tool developers then carefully calibrate the mental models embedded in the autonomous tool to fit with the mental models of the designers.

Software interface design at Adobe Labs. Interface designers today make extensive use of machine learning to improve interface designs. For example, researchers at Adobe Labs created tools intended to control complex user processes.[13] In particular, visual designers wanted to be able to control procedural models that render complex shapes (such as trees and bushes) by growing them artificially from digital seeds into mature plants. Designers had difficulty harnessing these models because the array of approximately 100 parameters controlling such a growth process had to be manipulated in unison, thereby making it an incredibly complex problem space. Machine learning provided a solution, enabling Adobe Labs' developers to reduce this high-dimensionality problem to a three-dimensional space comprehensible by human designers. Moreover, the three-dimensional space was controllable through three slider bars. Using this intuitive interface, designers can more easily configure the model to match a given image. The example shows that autonomous tools do not have to correspond precisely to the mental models of humans. Instead, they often provide an expressive but low-dimensional interface. Humans learn through interacting with this interface, and the machine and the human both participate in learning. The interface amplifies the ability of a human designer to explore a large design space. In this design process, the autonomous tools create an interface a designer can use to generate alternative outputs, as in Figure 2, loop 1. Through practice, designers learn how to control the outputs (see Figure 2, loop 2). Over time, the designer's experience can be used to refine the interface, as in Figure 2, loop 3. In such user-interface design, the machine-learning system begins with the goal of reducing the dimensionality of the interface from 100 dials to three slider bars. Although the mental model of the human can never be entirely aligned with the underlying mental model embedded in the tool due to the limits of human cognition, the new interface provides a level of abstraction necessary for effective learning.

Designing *Landscapes* at Ubisoft. Tools have a long track record in video game development. Algorithmically generated content may include a variety of game elements, including textures, buildings, road networks, and component behaviors like explosions.[7] In extreme cases, autonomous tools are able to generate large parts of the game content that only later are

Figure 4. Procedural generation in *Ghost Recon Wildlands*; source: Ubisoft.

combined with specific handcrafted elements. Hence, the interplay of automated and manual generation of content is crucial to game development, as humans are looking for a rich and unique experience, and undirected automated generation could lead to results that are not perceived as authentic. Ubisoft's *Ghost Recon Wildlands*, an action-adventure game originally published in 2017, is an example in which designers used autonomous tools to generate the game environment.[12] Designers handcrafted elements in the game environment while algorithms procedurally generated much of the background content. In this process, the tools would generate, say, large amounts of detailed terrain; the Figure 4 screenshots show how the terrain evolved as a road network was added procedurally, based on a few waypoints on a landscape. The designers would then modify the terrain further and create extra detail.

Some areas of the game environment were still generated in a traditionally manual fashion. The combined process required selecting appropriate tools and models that would align with the game idea in a way that was shared by a team of Ubisoft designers and developers. This example of "hybrid" development highlights how, although the tool autonomously generated significant portions of the design, designers still had a significant role in the design process. In such a "hybrid" model of autonomous design, the tool and the designer jointly generate the design in a given problem space (see Figure 2, loop 1); based on feedback generated by the tool, designers make adjustments and design decisions (see Figure 2, loop 2); and the team learns holistically from the experience of using the tool, reflecting on the alignment of their mental models with the outcomes of the use of the tool (see Figure 2, loop 3).

Artificial intelligence design and Atari games. Many researchers today engage in designing artificial intelligence solutions, using machine learning in their solutions. For example, researchers recently created an artificial intelligence system to play Atari games. The experimental system was a deep convolutional neural network with inputs wired to a video game display and outputs wired to a joystick controller.[10] The system used a reinforcement-learning algorithm to train itself. After training, it scored as well or better than humans on 29 out of 49 Atari games. But some games proved challenging for the algorithm, and games that require a player to acquire objects early in the game that will prove useful only near the end were especially difficult for the algorithm. Taking such decisions across long time scales is more difficult to learn than are subsecond reactions to, say, attacking enemies. As a result, the designers of the system made modifications to the training algorithm that significantly increased its performance on difficult games, though they still require hundreds of hours of training.[8] In this case, when the algorithm is exposed to gameplay events, it learns, as in Figure 2, loop 1). When the machine fell short on certain games, the designers adjusted the training regimen (see Figure 2, loop 2). In creating the system, the designers first created an environment tools explore and receive feedback on, similar to the way humans interact with the physical environment.

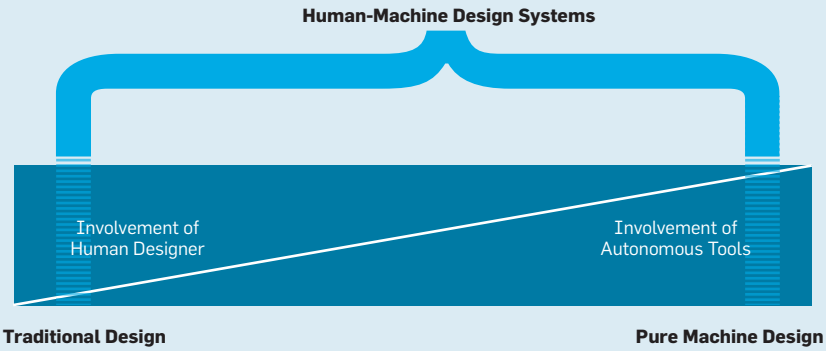
Tool developers then carefully calibrate the mental models embedded in the autonomous tool to fit with the mental models of the designers.

At a higher level of abstraction, the process can be viewed as part of a meta learning process in which humans create autonomous machines, monitor their progress, and iterate across multiple configurations of the machines while ultimately confronting the limits of both machine and human intelligence (see Figure 2, loop 3). The shortcomings of the algorithm indeed pose a challenge some researchers argue are best addressed through techniques developed in cognitive science.[9] Even the use and development of autonomous systems are examples of triple-loop learning, as these systems need to be designed, monitored, and improved by humans.

Designers and Triple-Loop Design Activities

Traditional designers intentionally craft artifacts by applying their deep knowledge of materials and tools, moving them forward toward a preferred, future condition.[11] However, autonomous tools change the role of designers, including focus, activities, and required skills. Designers are increasingly focused on manag-

Figure 5. Shifting control in design processes.



New design practices.

Design Practice	Example	Description
Framing	Parameterization	Designers have a deep understanding of the software tool and its parameters, as well as some understanding of the consequences of setting specific parameters; and They formulate hypotheses with regard to what sets of inputs will have the desired design consequences in lieu of carrying out the entire design process in an incremental, iterative, primarily manual fashion.
Evaluation	Process Analysis	Designers evaluate the overall design outcome, investigating sources of misalignment, as in assumptions embedded in the tool; and They formulate hypotheses about the process and test whether they hold.
Adjustment	Modifying Algorithms	Designers continuously align their mental models with mental models embedded in the autonomous tools; They consider how changes in the constraints and propensities of the tool may require changes in their mental models in terms of assumptions and goals; and They consider how changes in assumptions and goals may require changes in the mental models embedded in the autonomous tools.

ing tools—and their embedded mental models—and understanding the often-surprising behaviors of tools as they generate design artifacts. This new type of designer needs a better understanding of the tools, in addition to a detailed understanding of the underlying anatomy of the artifact to be designed. The locus of control of the design process is moving away from the designer toward the tool and its underlying model. An important causal force behind the tool is the tool designer who defines and implements the algorithmic choices. The tool designer thus creates the initial version of the mental model embedded in the tool, a model that will change as the tool itself learns. As illustrated in our examples, there can be different degrees to which control shifts toward the tools and away from the designer (see Figure 5).

Rather than incrementally build and modify design artifacts, designers become engaged in new design practices that fall into three categories: framing, evaluation, and adjustment.

Framing. "Framing" occurs as designers, based on their mental models, construct their understanding of the problem situation or task and thus how the tool, with its underlying, embedded mental model, should be used, thereby making decisions about the solution space. The actual design activity is thus informed by both the mental model of the designer and the mental model embedded in the autonomous tool.

Framing in the examples outlined earlier notably involves specification of varying sets of inputs that can include numerical and non-numerical variables, thus enabling the tool to do its work. This is the process of "parameterization," which requires a deeper understanding of the tool, as well as an intuitive understanding of both the problem space being worked on and the solution space of the tool so hypotheses can be formulated with regard to what sets of inputs will have the desired design consequences. Parameterization thus precedes the actual design process (see Figure 2, loop 1) and follows the evaluation of the design product.

Evaluation. Once the autonomous tool has generated outcomes, these outcomes must be evaluated to inform decisions about further design actions (see Figure 2, loop 1), as well as to inform the mental model of the designers (see Figure 2, loop 2a) and the mental models embedded in autonomous tools being used (see Figure 2, loop 2b). While loop 1 activities lead to a different use of the tool through, say, a different set of input parameters, loop 2 activities lead to changes in mental models that affect future design decisions.

As parameters can be changed and various design alternatives explored, autonomous tools allow more iterations of the design outcome and thus for experimentation. For instance, Ubisoft's video game *Ghost Recon Wildlands* presents an experience to users that is possible only because a relatively small team of designers could experiment with various computationally generated design outcomes.

Because the algorithmic processes of autonomous tools are typically complex, they tend to overwhelm humans' bounded cognitive abilities. It is difficult for human designers to predict what the tools will produce, so they must be able to evaluate the design products generated by the tool. Such evaluation may range from specific aspects of the outcome (such as elements in the game space of a video game) to some holistic outcome (such as in the process of generating the layout for a semiconductor chip). Once a tool has been run, and has generated outputs, designers evaluate the

outputs in a way that leads to new hypotheses with regard to what sets of input parameters should be tested in the next batch of experiments.

Adjustment. Evaluation by human designers can lead to the adjustment of parameter values (see Figure 2, loop 1) or even to changes in the mental model embedded in the autonomous tool, resulting in changes in the algorithms used; moreover, it might also change the mental models of human designers in terms of goals, cognitive rules, and underlying reasoning. Changes of the mental model embedded in the autonomous tool could change the tool's constraints and propensities and require changes to the mental models of designers; likewise, changes in the mental models of designers could require changes to the algorithms and thus the mental model embedded in the tool. Following each experiment, designers might thus have to continuously reconcile their mental models with the counterpart models embedded in the autonomous tool (see Figure 2, loop 3).

In order to change the mental model embedded in an autonomous tool, designers have to modify the underlying algorithm. The original mental model embedded in the tool—the one implemented by the tool designer—can thus evolve over time.

Competencies related to these design practices become critically important for achieving complex design outcomes. Having a detailed understanding of the designed artifact, as well as of the consequences of specific local decisions, becomes less important. This explains why, in the context of, say, chip design, we see software engineers displacing electrical engineers with a deep understanding of physical aspects of chip design. Because the design is increasingly mediated by software that needs to be parameterized and evaluated, designers' software skills become crucial; the table here outlines key implications in terms of emergent interrelated designer activities.

Some substitution of human design activity through autonomous tools is indeed occurring. To a certain degree, demand for specific, manual-type competencies in design professions, including software development, is decreasing, while the demand for skills focused on how to work with software tools is increasing. Organizations need to engage more effectively with new forms of autonomous tools supporting design processes. This is not simply a shift of tasks from humans to machines but a deeper shift in the relationship between humans and machines in the context of complex knowledge work. The shift puts humans in the role of coaches who guide tools to perform according to their expectations and requirements (see Figure 2, loop 1) or in the role of laboratory scientists conducting experiments to understand and modify the behavior of complex knowledge artifacts (see Figure 2, loop 2 and loop 3).

The Road Ahead

Engaging with autonomous tools requires reshaping the competencies designers need. Designers envision certain results and thus need to interact with autonomous tools in ways that help them realize their design vision. At the same time, the use of autonomous tools opens unprecedented opportunities for creative problem solving. Consider the example of video game production, where autonomous tools are increasingly able to procedurally generate artifacts of a scope and scale that was not possible in the past. Future designers will constantly be challenged to rethink their mental models, including their general approach to design. The continuous reconciliation of mental models embedded in both designer cognition and their tools is an extension of traditional design processes that involve artful making where human actors gradually adjust their mental models to converge on solutions.[5]

The proposed three-loop model contributes to the ongoing debate on how artificial intelligence will change knowledge work, challenging knowledge workers to operate at a different level. Designers may become increasingly removed from the actual artifact but still use tools to create artifacts of a complexity never imagined before.

Acknowledgments

This material is based in part on work supported by the National Science Foundation under grants IIS-1422066, CCF-1442840, IIS-1717473, and IIS-1745463.

References

1. Allen, R.B. Mental models and user models. Chapter in *Handbook of Human-Computer Interaction, Second Edition*, M.G. Helander, T.K. Landauer, and P.V. Prabhu, Eds. North-Holland, Amsterdam, the Netherlands, 1997, 49–63.
2. Argyris, C. The executive mind and double-loop learning. *Organizational dynamics 11*, 2 (Autumn 1982), 5–22.
3. Argyris, C. Teaching smart people how to learn. *Harvard Business Review 69*, 3 (May-June 1991).
4. Argyris, C. *Double-loop learning.* Chapter in *Wiley Encyclopedia of Management*, C.L. Cooper, P.C. Flood, and Y. Freeney, Eds. John Wiley & Sons, Inc., New York, 2014.
5. Austin, R.D. and Devin, L. *Artful Making: What Managers Need to Know About How Artists Work.* Financial Times Press, Upper Saddle River, NJ, 2003.
6. Brown, C. and Linden, G. *Chips and Change: How Crisis Reshapes the Semiconductor Industry.* MIT Press, Cambridge, MA, 2009.
7. Hendrikx, M., Meijer, S., Van Der Velden, J., and Iosup, A. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications 9*, 1 (Feb. 2013), 1.
8. Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. In *Proceedings of the Fifth International Conference on Learning Representations* (Toulon, France, Apr. 24–26, 2017).
9. Lake, B., Ullman, T., Tenenbaum, J., and Gershman, S. Building machines that learn and think like people. *Behavioral and Brain Sciences 40*, E253 (2017).
10. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., and Petersen, S. Human-level control through deep reinforcement learning. *Nature 518*, 7540 (Feb. 2015), 529–533.
11. Sennet, R. *The Craftsman.* Allen Lane, London, U.K., 2008.
12. Werle, G. and Martinez, B. *Ghost Recon Wildlands: Terrain tools and technologies.* Game Developers Conference (San Francisco, CA, Feb. 27–Mar. 3, 2017); https://666uille.files.wordpress.com/2017/03/gdc2017_ghostreconwildlands_terrainandtechnologytools-onlinevideos1.pdf
13. Yumer, M.E., Asente, P., Mech, R., and Kara, L.B. Procedural modeling using autoencoder networks. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, Nov. 11–15). ACM Press, New York, 2015, 109–118.

Stefan Seidel (stefan.seidel@uni.li) is a professor and the Chair of Information Systems and Innovation at the Institute of Information Systems at the University of Liechtenstein, Vaduz, Liechtenstein.

Nicholas Berente (nberente@nd.edu) is an associate professor of IT, analytics, and operations in the Mendoza College of Business at the University of Notre Dame, Notre Dame, IN, USA.

Aron Lindberg (alindberg@stevens.edu) is an assistant professor of information systems in the School of Business of Stevens Institute of Technology, Hoboken, NJ, USA.

Kalle Lyytinen (kalle@case.edu) is a Distinguished University Professor and Iris S. Wolstein Professor of Management Design at Case Western Reserve University, Cleveland, OH, USA.

Jeffrey V. Nickerson (jnickers@stevens.edu) is a professor of information systems and the Associate Dean of Research of the School of Business at Stevens Institute of Technology, Hoboken, NJ, USA.