Computers and Creativity

By Molly Mielke

> *True creativity and invention, which are the seed of innovation, come from people and they come from the stories of people. They come from their backgrounds, their passions, what moves them, the things that worry them, the things that are their dreams.*
>
> — Frank Moss,
> *The Sorcerers & Their Apprentices*

Introduction

The value of computers is not inherent; it is what we are able to do with them that makes them valuable. But what we can do with computers is often limited by the depth to which we are able to think creatively, translate these thoughts into computationally articulated work, and then share that work with others. For this reason, digital tools that foster creativity and collaboration hold immeasurable power. So how can we push digital creative tools to their full potential as co-creators, thus harnessing the full power of creative thought and computational actualization to enable human innovation? Ultimately, I will be arguing that to foster optimal human innovation, digital creative tools need to be interoperable, moldable, efficient, and community-driven.

Early creative tools

Beginning with the early days of the workplace computer in the 1950s, computers began as purely logical room-sized execution machines that operated on paper and punch cards. A human would feed the computer tasks in the precise way the computer understood, and the computer would perform a desired function for the operator. Usually, those functions were basic mathematical operations. This was revolutionary for its time and embodied a task-based relationship between computer and machine that continues to define how we operate computers today.

However, a new way of thinking about these machines began to emerge in the late 1980s. From the punch card came the spreadsheet, which eliminated the need to provide physical inputs and allowed us to utilize math in a new way. Meanwhile, creative applications for computers began to emerge. As the *Victoria and Albert Museum* present in "A History of Computer Art," "In the 1950s, many artists and designers were working with mechanical devices and analogue computers in a way that can be seen as a precursor to the work of the early digital pioneers who followed. One of the earliest electronic works in the V&A's collection is 'Oscillon 40' dating from 1952. The artist, Ben Laposky, used an oscilloscope to manipulate electronic waves that appeared on the small fluorescent screen." Throughout the history of digital tools, a common theme is the technological advancement spurred by artists pushing the limits of the existing tools and creating their own adjacent tools to better serve their needs.

Historically, the primary strains of creative tools have been text editors, spreadsheets, HyperText,[1] and Computer Aided Design (CAD) software. For the purposes of this essay, "digital creative tool" will refer to tools that foster creative thought in their human users, whether that be a design, writing, or multimedia tool. With the introduction of Superpaint,[2] one of the first image editing programs in 1984, the concept of the digital creative tool began to take its initial form—pioneering conventions that are still in use today such as the digital canvas and the tooling menu. However, it is worth reconsidering aspects of these historical conventions in the interest of fostering greater human creativity using digital tools.

[1] Hypertext was first coined by Ted Nelson and refers to text that links to other texts.

[2] Superpaint was created by Xerox PARC employee Richard Shoup, among others.

The genesis of many original creative tools were inspired by the work of thinkers such as Douglas Engelbart, who published a vision of tools empowering humans in his pioneering 1962 paper *Augmenting Human Intellect: A Conceptual Framework*. As could be surmised, his goal was to use digital tools to augment human intelligence and thus boost our collective intellect. Engelbart's work, along with many other designers and programmers such as Ivan Sutherland (Sketchpad[3]), J.C.R. Licklider (Intergalactic Computer Network[4]) and Alan Kay (Dynabook[5] and OOP[6]) clarified a new approach in which computers weren't merely executors, but "joyful" machines that could expand human thought itself (Engelbart). As Engelbart put it, "tools… will serve as new media of expression and inspiration to creativity." Engelbart was not alone in this thinking and inspired other computer scientists like Alan Kay to study how computers could amplify imagination. While these early manifestations of creative tools unlocked pioneering ideas on the relationship between humans and machines, they were largely conceptual in nature. Engelbart and Kay's work hinted at a future of creative tools built to serve the human mind's creativity, but were missing the financial support from technology companies to invest further in the relationship between humans and machines as opposed to investing primarily in machines themselves.

[3] Sketchpad was the first CAD program to have a complete graphical user interface.

[4] The Intergalactic Computer Network was a computer networking concept similar to today's Internet that was imagined by J.C.R. Licklider as an "electronic commons open to all."

[5] Dynabook was Alan Kay's vision of a "personal computer for children of all ages."

[6] Object-oriented programming, or OOP, is a programming paradigm pioneered by Alan Kay using the concept of "objects" containing data and code.

Other visions of creating space for creativity and human thought can be found within the bordering space of software development. It was the people that created Apple's OpenDoc[7] (1997) that pioneered one of the first steps towards standardization and collaboration between creative tools. As Kristi Coale states in "Closing OpenDoc-a Great Leap Backward," "In its heyday, OpenDoc was seen as the future of document creation. No longer were users limited by the capabilities of an application in making a document; they could include video, audio, and spreadsheet input created in other applications and tie them into a single large document." This approach to standardizing digital files offered a new opportunity for far more people to use computers in creative ways. OpenDoc did not necessitate the selection of a single media type and instead allowed for mixtures of different kinds of media. While there was still far more work to be done for OpenDoc to reach true interoperability, the software established the idea of making digital creative work tool agnostic. However, it is worth noting that the structure of OpenDoc was still adhering to replicating the formation practices of physical work. The project as a whole was eventually discontinued by Apple due to an unsuccessful business model.

However, another of Apple's earlier projects called the Hypercard[8] (1987) found greater success and proved the potential of a moldable approach to software development. As Samuel Arbesman states in "The forgotten software that inspired our modern world," "Bill Atkinson, its developer, described HyperCard as 'an erector set for building applications. Simply put, you could build your own software using HyperCard, with each program made up of 'stacks' of 'cards'. Each card could contain text and images, as well as interactive elements like buttons, with the ability to interconnect between other cards." HyperCard was a tool for making tools, and was unique in its focus on encouraging open collaboration between users. This approach helped foster a powerful community of people who contributed to the moldable tool's development and evolution.

[7] Apple's OpenDoc was a component-based framework standard for compound documents, inspired by (and intended as an alternative to) Microsoft's Object Linking and Embedding (OLE) technology.

[8] Apple's Hypercard was a software application and development kit for Apple Macintosh and Apple IIGS computers, and was one of the first hypermedia systems predating the World Wide Web. HyperCard pioneered the first wiki, which in turn served as the inspiration for Wikipedia.

Digging deeper into the history of creative tooling communities reveals the lasting prevalence of the communities surrounding Flash[9] and Actionscript[10] in the early 2000s. Flash and Actionscript were popular, easily manipulated, low-bandwidth tools that created the first interactive experiences on the web. As Joshua Granick states in his article on ActionScript 3.0, "In 2001, the popularity of Flash and ActionScript continued to grow as artists and developers discovered the infinite possibilities of an expressive web platform." Before being acquired by Adobe, the tool itself was deeply embedded in its community's contributions, despite never going so far as to be open-source. This relationship between Flash and its users was most clearly seen in the online discussion forums, which pioneered the expectation that community members would frequently share tips, feedback, and projects with one another. This culture, combined with the abstracted and efficient nature of the scripting language, fueled Flash and Actionscript's widespread adoption and quickly reshaped the web to become interactive and animated. Actionscript enabled projects like a map generator, a prebuilt ActionScript 3 library for integrating Flash games, the Starling Extension Particle System, and many other projects (Github, "Awesome Actionscript"). Powered by community, Flash serves as a proof point that widespread ownership over an abstracted and moldable creative tool holds immense potential to enable creativity in people at scale and spur organic user growth and adoption as a byproduct.

More broadly, the role of collaboration in digital creative tools has been complex and multifaceted. The shift to computing in a network has redefined how we think about collaboration using digital tools. Beginning with the birth of the internet with the help of those such as Ted Nelson,[11] we quickly moved on to a period of mobile technology proliferation. Collaborative software as we know it was born soon after in the form of Google Sheets, the first simultaneous multiplayer software. The state of collaborative

[9] Flash was an authoring program originally created by Macromedia and used to create vector graphics-based animation programs, usually for the web.

[10] ActionScript was an object-oriented programming language by Macromedia used primarily to develop websites and software using the Adobe Flash Player platform.

[11] Ted Nelson is an American pioneer of information technology, philosopher, and sociologist. He coined the terms hypertext and hypermedia in 1963 and published them in 1965. Nelson also introduced the term intertwingularity.

software today is a testament to the progression of tools for creativity, as well as the room we still have left to help them reach their full potential.

The origins of digital creative tools show that the most boundary-pushing and high-potential tools were often interoperable, moldable, community-driven, abstracted, and efficient, thus actualizing creativity within the tool itself. Upon review, it is clear that the fundamental human-computer interaction principles of the past have remained unchanged, such as the direct manipulation of graphical objects, the mouse, and windows. However, it is also evident that our expectations for a computer's capacity to understand and serve us has expanded considerably (Brad Myers, "A Brief History of Human Computer Interaction Technology"). We will now examine this gap between our historical-ideals-driven expectations and the contemporary reality of digital creative tools.

Creative tools today

Today, computers have become a melded version of both a task-based execution machine (calculations, programming) and an intelligent creative co-creator (writing and design tools). As João Miguel Cunha states in "Generation of Concept-Representative Symbols," we are in the process of "shift[ing] away from using computers as tools and see[ing] them [instead] as partners, as well as mak[ing] computational co-creativity applications available to the general public." However, in recent decades, this shift has been slow and undefined, only noticeable to the computer-using public in new consumer softwares boasting mainly efficiency upgrades.

So if this shift to co-creation is happening (albeit slowly), where exactly is it taking place and who is leading the charge? Despite their brief moment in the internet limelight, creative computing communities have failed to maintain power over the software they operate. A close examination of the digital creative tools landscape today reveals that most innovation in this space has shifted away from the computer users (as seen through Actionscript and Flash) to instead by owned by the heavily funded R&D departments of the dominant Silicon Valley technology companies. As Alexis Madrigal states in "Silicon Valley Abandons the Culture That Made It the Envy of the World," "Quantitative research suggests that big companies do different kinds of R&D than their more modest counterparts. Instead of coming up with new products, they come up with process improvements. 'If the nature of innovation is distorted toward selling to an incumbent,

you're going to get more feature-driven innovation rather than systemic disruption.'" This statement is directly reflected in digital creative tools' lack of concept-level innovation in the past ten years. We see this further exemplified through tools looking and operating very similarly to how they did at their founding.

However, it is worth noting the significant advancements that have been made within the existing creative tooling structures. Integrating collaboration features into creative tools has been a major development in the past decade and initiated the creation of new spaces for cross-functional creative work. In part as a result, digital real-time collaboration applications such as Figma have seen great success. The increasing prevalence of these collaborative creative tools also means that functional interoperability between them has become increasingly relevant to the work that we are all doing. Today's projects need to accommodate a multitude of different media types within cross-functional spaces, yet editing each piece is still largely constrained to whichever tool it was created in. This reality pinpoints an opportunity for both interoperability and further innovation in the collaboration-between-tools space.

The lack of concept development in digital creative tools poses another question: where have the technology industry's resources been funneled? A mere cursory look at technology news reveals the industry's heightened focus on artificial intelligence (AI)-fueled digital products. As Frederick Brooks explains in "The Computer Scientist as Toolsmith II," "A tremendous national investment has been made, over the course of more than three decades. Indeed, a large amount of this country's public investment in computer science research has gone into AI, compared with other promising opportunities. More serious even than the diversion of dollars was the diversion of the very best computer science minds of a generation, and much of the efforts of the very best academic laboratories."

While the opportunity AI promises is immense, the benefits of its innovation fail to articulate how they might be widespread enough to have a significant positive impact on human needs or creativity, in the case of our line of inquiry. Instead, these innovations seem to be solely in service of corporations. AI offers businesses the ability to make a single investment in a machine, as opposed to paying human workers indefinitely. For example, Alana Semuels of *Time Magazine* states in "Millions of Americans Have Lost Jobs in the Pandemic—And Robots and AI Are Replacing Them Faster Than Ever," "Now,

as automation lets companies do more with fewer people, successful companies don't need as many workers." This business-driven innovation fails to invest in the creativity of human beings themselves, instead seeking to streamline our capability to act as an execution machine. Directionally, this thinking is clearly in contradiction with the ideas of early digital creative tool pioneers such as Engelbart and Kay. Furthermore, this reality proves the need for us to reconsider our priorities for progress if we truly wish to foster greater human innovation.

Despite the technology industry's widespread investment in recreating human intelligence, the computer's capacity to be creative is still entirely dependent upon the human's ability to program that prescriptive creativity step-by-step into a machine. As Tony McCaffrey of "There Will Always Be Limits to How Creative a Computer Can Be states," "The fastest modern supercomputer couldn't list or explore all the features of an object/thing even if it had started working on the problem way back in the 1950s. When considering the Obscure Features Hypothesis for Innovation,[12] which states that every innovative solution is built upon at least one new or commonly overlooked feature of a problem, you can see how AI may never advance enough to take the jobs of Chief Innovation Officers." While there are situations in which computers are able to figure out the steps needed to get from state A to state B, it is still within the limitations of pre-human-mandated decomposition. This means that the actions we take with computers are steps that we have already broken down in our head in order to achieve an end result. This is often a highly repetitive process, and makes clear that there is room for computers to become better co-creators in order to make space for human creativity.

Acknowledging that computers themselves are not inherently creative should not come as a surprise. Instead, this truth identifies an opportunity for computers to more fully assume the role of co-creator — not idea-generator, but actualizer. Furthermore, it is worth acknowledging that human creativity and innovation is not a solitary sport. In fact, most great digital innovations are the product of a commitment to collaboration and the melding of many different perspectives.

Facilitating collaboration in a digital tool is a key multiplier for creativity. When you bring people together to create in the same space (whether it be digital or physical), they are able to learn from and build upon each other's work through the process of

[12] The Obscure Features Hypothesis (OFH) states that all innovative solutions are built upon at least one overlooked feature of the problem at hand.

collaborative knowledge creation. This reality again highlights the pertinence of functional interoperability since the cross-pollination of digital work between different tools is necessary for collaboration to function as smoothly as possible. Fostering both creativity and collaboration in a co-creation tool harnesses the power of computation and the potential of multiple human perspectives.

As Steve Jobs once explained during an interview in “Memory & Imagination: New Pathways to the Library of Congress,” “that’s what a computer is to me... it’s the most remarkable tool that we’ve ever come up with, and it’s the equivalent of a bicycle for our minds.” This framing steers our attention away from automating digital tools to become creators themselves. Instead, we can reconfigure our goal towards aiming to construct software intelligent enough to figure out the steps needed to produce a desired outcome in service of the human or group's broader creative vision. As Shan Carter and Michael Nielsen explain succinctly in “Using Artificial Intelligence to Augment Human Intelligence,” “Intelligence Augmentation (IA), is all about empowering humans with tools that make them more capable and more intelligent, while Artificial Intelligence (AI) has been about removing humans fully from the loop.” Using this perspective, my argument will propose several ways that we can refocus on Engelbart, Carter, Nielsen, and Job’s shared vision of augmenting human intelligence using digital creative tools.

Significance

By reviewing the history of digital creative tools and the ways computers foster creativity and collaboration today, we can begin to synthesize the requirements for co-creation tools. Focusing on enabling creativity optimizes for the type of innovation only humans are able to perform using a computer, which in turn offers the potential to reshape humanity in new, and hopefully better, ways. While I am not asserting that technology holds all the solutions, I am arguing that when appropriately applied, technology can solve large problems, enable creativity, and change lives. So we return to our question: how can digital co-creation tools augment human creativity and collaboration?

Standardization

Taking a step back, we might consider the fundamental form in which creative work is created, recorded, and stored. Today, there is next to no standardization between digital creative tool files. While there are sometimes ways to convert one tool's file type to another, the process is tedious at best, unusable at worst. The lack of interoperability[13] between creative tools means that all work created within a tool is confined to the limitations of that tool itself, posing a hindrance to collaboration and limiting creative possibility. If tools are meant to amplify the power of our brains and take over the mechanical aspects of human thought, limiting creation to a single piece of software's capabilities is clearly antithetical to creativity.

Returning to our comparison between the development and design communities, the development community's prioritization of functional standardization has led to far greater leaps in collaboration and wider scale public ownership over innovation. A culture of building things with the express purpose to be shared and built upon has created a baseline standard of interoperability within these communities. This has enabled scaled contributions on Github projects such as Microsoft's open-source VS Code text editing software (Nick Kolakowski, "Top 10 Most Popular Open Source Projects on GitHub"). Interoperability has been key to sufficiently building upon each other's innovations and contributions, while spurring the field of development's speed of innovation.

The ongoing efforts of companies such as GitHub to incentivize the creation and maintenance of open source software and programming languages is worth contextualizing. As Sidney Fussell explains in "The Schism at the Heart of the Open-Source Movement," "technology firms rely on open-source licensing, a legal framework that lets users borrow ideas and pool together the insights and labor of volunteer developers. GitHub is itself built on open-source tools, and sometimes uses code hosted on the platform to improve itself." However, the reality of open-source projects themselves begs reexamination. As Nadia Eghbal writes in her book *Working in Public,* "One study found that in more than 85% of the open source projects the researchers examined on GitHub, less than 5% of developers were responsible for over 95% of code and social interactions." While there is impressive work being done to

[13] In this context, interoperability refers to the ability for different computers to connect and exchange information.

standardize computing and make development more interoperable and accessible, it still lacks the distributed ownership that incentivizes widespread interoperability.

Nonetheless, programming languages themselves have seen moderate success in standardization. The partnership between R, a free software environment for statistical computing and graphics, and Python, an interpreted, widely-used programming language, serves as a poignant example of the global value derived by interoperability. As Dan Kopf states in “R and Python are joining forces in the most ambitious crossover event of the year for programmers,” “[Python] will partner with RStudio [R]… The main goals…are to make it easier for data scientists working in different programming languages to collaborate, and avoid redundant work by developers across languages.” Increased collaboration and efficiency directly exemplify the benefits of interoperability at scale.

However, it is also worth examining the issues associated with interoperability. Interoperability can often slow down improvements and lead to inconsistent adoption of open standards. A poignant example of this can be found in the lack of universal browser compatibility for HTML/CSS features, which adds unnecessary complexity to web development work. Another consideration is the risk that standardization may commoditize a set of tools, thus diminishing the innovation made possible by healthy competition. An example of this can be seen in the state of web browsers today. When web developers do not properly support all the different web browsers, they inadvertently drive adoption towards their own tools of choice. This is exemplified through the contrasting support and usage of Google’s Chrome and Mozilla's Firefox. However, these considerations are largely the result of building upon a highly functional but extremely fragmented structure of programming that only implemented consistency and standards after much of the architecture of the web had already been built. Digital creative tools, however, are at a different place in their evolution. Defining interoperable standards between these tools would be altogether new and informed by learnings from similar implementations in other industries. Standardization would amplify the power of each tool and vastly expand the possibilities of digital creation.

Taking further inspiration from the development example, we can begin to visualize the possibilities of standardization if employed for digital media. We might imagine interoperable source files being stored in a repository structure that could then be opened and modified by any creative tool of choice. This would require standardization of

file types, metadata, and a single substrate. Source files might take inspiration from Extensible Markup Language, or XML files today. XML files are unique in their being both human and machine-readable, lightweight, and widely recognized by almost any software tool.

This concept would not only open doors to further collaboration at scale—it would also effectively turn the computer into a breeding ground for human innovation. Standardization would fundamentally change the tide of digital creative tools for the better by allowing in more collaborators, making space for greater tooling innovation, and expanding a project's creative constraints beyond any one tool itself.

Moldability

Beyond standardization, there is still more to be done within the confines of the digital tool to make it a better co-creator with its human counterpart. Today, the way that we use creative tools is relegated to the boundaries drawn by the software company who created the tool. However, when considering the ingredients needed to facilitate creativity, a common theme that arises is the need for software moldability, or the ability for the user to tailor their software to better address the problem they are trying to solve. I will argue that making tools moldable to their users' preferences while cultivating communities that inspire and help members shape their own tools is the next logical step for computers to become better co-creators.

Returning to Engelbart's guiding principle, computers have the power to change and expand human thought. But to do so, the software must adapt to suit the user's unique thought process. This idea shines a light on the importance of a tool's moldability, as measured by how easily the software can be customized to the average non-programmer's needs. As Mohamed Fayad and Marshall P. Cline state in "Aspects of Software Adaptability," "Flexibility means it is easy to change the system's capabilities in kind. For example, taking something that was a graphical system and making it sensory-or sound based. Flexibility is often harder than extensibility, especially when on-the-fly changes are desired." Fayad and Cline highlight that there is a clear correlation between how flexible the software's codebase is and the user's experience of how moldable the tool is to their creative process.

A moldable development environment called “GToolkit” exemplifies a similar correlation within the field of development. As Tudor Girba states in his presentation about GToolkit called “Molding Objects with Moldable Tools:” “these tools can be customized live with very little effort. The low moldability cost, and we often talk about minutes to adapt the tools, opens new ways to approach understanding code and runtime through live objects.” In this context, moldability refers to the tool’s readiness to accommodate many different tasks in many different ways, without forcing the user to conform to any set processes defined by the software.

A common argument against moldability is the increased cost of maintaining integrations or modifications that exist on top of a perpetually changing piece of software. This is a real issue, as evidenced by the fragility of plugin ecosystems such as those of the free content management system, Wordpress. However, this points to the fact that there are many different types of moldability to consider. Moving beyond building on top of existing software, we can begin to imagine what a piece of software could look like if it itself was moldable, or built to be modified by the user. As the Victoria and Albert Museum explain, an interesting example of this can be seen in the early 1960s: “by writing their own programs, artists and computer scientists were able to experiment more freely with the creative potential of the computer.” However, the evolution and global spread of personal computing since then has meant that this capability is no longer limited just to programmers. By simplifying and contextualizing the process of shaping the tool to their needs, the user is able to make the tool exactly what they need to foster their unique creativity.

No tool exemplifies moldability better than music digital audio workstations, or DAWs. DAWs such as Apple’s Logic Pro and Ableton Live are unique in their extensive ability to be customized and accommodate plugins, hardware, and files of all types. In part for these reasons, DAWs are widely loved by their users and serve as a shining example of software making very little assumptions about the user’s creative workflows. Instead, DAWs accommodate numerous paths to reach a desired outcome, in addition to providing the resources and community to teach the user how to create any missing feature themself. Similarly within creative tooling, Robofont also comes to mind when thinking of moldability. The “fully featured font editor with all the tools required for

drawing typefaces" was designed to be built on top of, with customization options and integrations for a multitude of different scripting languages at its core.

Moldability as seen through scripting languages can also be seen exemplified through Actionscript in its early years. Interjectable scripting languages effectively removed the steep learning curve that users would need to climb to fully understand how to build their own tools themself. Actionscript provided users with abstracted, easy-to-understand coding building blocks, accompanied by a community of other users sharing their creations, learning, and pushing the bounds of the tool itself. Within the broader context of the community as a whole, the object-oriented programming language inspired a sense of play and experimentation in the user that served as a potent incentive to create. As Bill Gaver wrote in *Designing for Homo Ludens:* "The designer's role in this is not like that of a doctor, prescribing cures for people's ills; nor is the designer a kind of servant, developing technologies that people know they want. Instead, designers should be provocateurs, seeking out new possibilities for play and crafting technologies that entice people to explore them."

ActionScript and Flash attest that the success of a moldable tool often comes down to the tool's ability to inspire exploration and spur a vibrant tooling community. Pioneering users that share their creations and anecdotally attest to the tool's viability and promise can be seen as toolmakers. Toolmakers are essential to bringing moldable software building blocks to the average user—the toolmaker's shared creations serve as invitations for the average user to join and create something themself. However, it is also vital that the tool's moldability is abstracted enough to be widely accessible—usually meaning that the interface should be controlled visually and require no technical knowledge. Positioning a tool in this way allows toolmakers to lead the way and inspire average users to mold the tool themself.

Making tools adaptable to their user's thought process is the first step in facilitating more human creativity with computers. Baking flexibility into software is also in some ways advantageous for the software's business. Moldable tools allow the user to build for their own needs as opposed to relying solely on the software company to deliver. This process also fosters a sustained sense of personal ownership and loyalty to the tool. Furthermore, the communities that emerge out of moldable tools demonstrate the creativity and collaboration that come as a result of ownership being shared between the

software creators and the software users. Allowing the tool to be customized and built within a collaborative community serves as a powerful example of how we might facilitate greater creativity with these tools. This concept also suggests that we consider alternative ways in which digital tools could abstract inefficient areas of the creative process.

Abstraction

In considering how computers might become better co-creators and actualizers, we can begin to identify more granular ways that these machines might become more classically "efficient" in facilitating and keeping pace with human creativity. However, it is important that we first acknowledge the tension between efficiency and creativity. While on the surface the two dimensions may seem at odds with one another, this does not have to be the case. The computer's execution skill set lends itself to minimizing the need for any sort of repetitive or monotonous work that might hinder the creative process. This truth refocuses our attention on fostering creative thought (such as ideation) as opposed to linear thought (such as creating a functional architecture), seeing as the computer is already highly adept at accommodating the latter. As Tom Vanderbilt of Nautilus explains in "The Pleasure and Pain of Speed," "As we have shifted from manual typewriters to electric to, finally, digital tools… that technological speed bump has been eroding. He cites the research of Stanford University literary scholar Andrea Lunsford, who has examined freshmen entrance essays from 1917 until the present. While grammatical error rates have stayed the same, the length and complexity of the essays have dramatically increased. 'It's not that the kids of 1917 were stupider,' says Thompson 'It's just that their tools were getting in the way of their thought.'"

The challenge lies in training the computer to understand and execute a user's specific and often unique intent, with the goal of diminishing repetitive work and instead inspiring creativity and play. A simple example of this concept can be seen today in the form of predictive text editors. But we might also imagine how the same repetitive, logic-based principles could be applied to visual creative tools. Through this lens, the first step in computers becoming better co-creators would be to simplify repetitive workflows and accommodate logic in order to increase the efficiency of assembling in the creative tool.

Beginning with simplification, it becomes clear that this situation calls for abstraction, or the dumbing down of something technical in order to reduce complexity and omit unnecessary information. In abstracting repetition in the tool, we quickly realize that doing so requires accommodating a diversity of creative processes. While creative tools today offer components and other efficiency features, the contrast between the efficiency of creative tools and engineering tools is stark. Concepts such as abstract classes in development allow the programmer to apply objectified attributes while hiding the irrelevant details. In contrast, creative tools often require the construction of applied properties and details from scratch every time.

This repetition is a reflection of the constructive building model of canvas-based creative tools. The “elements-on-a-canvas” convention that we see at the core of creative tools such as Adobe Creative Suite products, Figma, and Sketch replicates the physical process of assembly that the digital tool has replaced. This approach has proven its worth: it allows for exploration and spatial orientation in a seemingly endless way. Replicating this process physically would require drawing possibly hundreds of screens by hand and then attempting to find the right one in a mess of papers. Despite the digitized version of this process being a significant improvement, iteration in digital tools today is still repetitive, time-consuming, and may deter potential creative contributors—especially in the simplest of use-cases. The scenario in which human-generated iterations mimic logical/data states proves particularly inefficient. While iteration on the digital canvas provides a broadly understood medium for creativity, it also necessitates humans act as machines to produce versions in the precise way the tool allows, which requires time and creative energy.

Considering how digital tools can be more efficient throughout the creative process from exploration to production, it becomes clear that moldability is key to accommodating the different stages’ needs. Beginning with the exploration stage within the example of interface design, a simple solution to abstract complexity and increase efficiency is to temporarily minimize options. Providing a simple, scalable interface with existing primitive elements and safe guides would help the user feel that making contributions is easy, simple, and reversible. This simplified tool would function like a lightweight scratchpad, providing both inexperienced and experienced users with predefined components and allowing them both to experiment rapidly. Offering

predefined elements (shapes, text, symbols) directly exemplifies how a digital tool can augment human intelligence by allowing the user to get into the creative flow faster and minimize unnecessary construction work.

However, this brings up a common objection by creative people against increased abstraction. Many believe that in minimizing options, you are effectively killing the creativity that would come from exploring during the process of assembly. While this argument is true, it is influenced by a comfortability with the processes of the past. The goal of increasing abstraction is to keep pace with human thought and provide the user with everything they need to articulate that thought without interruption. By simplifying the experience of using the tool, it invites in far more people than previously had access to the creative process. This fosters greater collaboration at the ideation stage, which is arguably the most important in order to accommodate a diverse range of perspectives.

Moving further along in the creative process to assembly, it becomes clear that the computer's execution skillset uniquely lends itself to generating a multitude of options for the human user to choose from. The process of assembly offers ample opportunity to tap into the power of human + computer co-creation by enabling the user to assign conditional logic[14] and dictate numerous variations at once. Returning to our example of interface design, we can imagine that as opposed to constructing surface-level properties that simulate how each state of an interface would appear, each state could be visually defined using conditional logic. Compositions would then be built from the sum of universal parameters. Logic would form an abstracted rendition of the digital output (code), thus providing a bridge between the creative and computational thought processes. Both variation generation and the accommodation of logic afford the human user greater time and mental capacity to focus on the creative choices that they are uniquely skilled to make.

Practically, incorporating variation generation and conditional logic in interface design is the difference between designing numerous different states of the same UI button versus designing one state and the corresponding logic buttons should follow. Logic shifts much of the needless production work back on the computer to follow linearly, as opposed to the human user working as a linear machine. As Frederick G.

[14] Conditional logic = { if/else made visual }

Linnemann and Carl Minichof state in “Logic Design System,” “The Logic Design System has been constructed to avoid built in hardware obsolescence."

The creation and widespread adoption of interface design systems can be seen as a first step towards streamlining ideation and increasing efficiency. As Clancy Stark of Figma found in his research on “Measuring the Value of Design Systems,” “when participants had access to a design system they completed their objective 34% faster than without a design system.” Incorporating logic into a digital tool is the next step in fully actualizing the computer’s potential as an execution-focused co-creator. This shift would also allow the user to spend less time on menial production work and more time on larger creative problems.

One level beyond logic in a creative tool can be seen exemplified in the concept of “programming by demonstration,” as coined by Bret Victor in his piece, “Magic Ink.” Returning to our example of interface design, the design process currently forces you to choose between two options: either learn a programming language (intimidating) or create mockups and have an engineer implement them (inefficient). Presently, the mockup process is the most common. However, Victor proposes another approach altogether called programming by demonstration; this technique involves teaching a computer what to do by exemplifying exactly what you want to have happen. This method means that the computer effectively infers and creates the logic needed to reproduce your demonstration. While yet to be fully realized within creative tools, this concept holds immense potential for further efficiency, while solidly situating the computer as co-creator.

Extreme examples of more fully abstracting logic and inputs can be seen in rare experimental applications of AI for creative tooling production generation. As Cade Metz of *The New York Times* reported on Jordan Singer’s explorations using the machine learning model GPT3 to generate specified code, “He fed the system a simple description of a smartphone app, and the computer code needed to create the app. The description was in plain English. The code was built inside Figma, a specialized design tool used by professionals like Mr. Singer. He did this a few more times, feeding the system several more English-language descriptions alongside the matching Figma code. And when he was done, GPT-3 could write such code on its own” (“Meet GPT-3. It Has Learned to Code (and Blog and Argue)”). This example demonstrates the purest form of co-creation

between human and machine. Abstraction and logic highlight how computers could take full ownership over execution and leave the human user to focus on the creative thinking that forms the whole.

Combining the concepts we've covered thus far brings us to the following conclusion: Computers have, since their inception, been a rigid tool that the human user has had to adapt to use. This limits what can be done with the tool because not everyone knows how to operate the machine in the precise way it requires, nor how to go about changing it to fit their needs. However, through standardization, moldability, and abstraction, we can dramatically expand the utility of computers while broadening their capacity to help more people solve their problems creatively.

Conclusion

But why does this matter? It matters because innovation is largely dependent on the human capacity to think creatively, and there is a strong argument to be made that technology's primary role is to speed up the creative process and catalyze innovation at a global scale. And yet as Lucien von Schomberg states in "Technology in the Age of Innovation," "in the current age… the concept of innovation is predominantly presupposed as technological innovation." This commonly-held viewpoint fails to capture the human creativity that presupposes innovation of any kind, even technological innovation. As the pioneering mathematician Richard Hamming put it in his book, *The Art of Doing Science and Engineering,* "The purpose of computation is insight, not numbers."

Interoperable, moldable, efficient, and community-driven digital creative tools hold immeasurable potential as co-creators with human beings. Tools of this type would lower the barrier to entry and make all users toolmakers and owners in an expanded definition of technological innovation. As Ted Nelson puts it in *Computer Lib/Dream Machines,* "everything is deeply intertwingled."

Bibliography

Arbesman, Samuel. “The Forgotten Software That Inspired Our Modern World.” BBC Future, BBC, www.bbc.com/future/article/20190722-the-apple-software-that-inspired-the-internet.

Austin, Robert D., and Lee Devin. “Research Commentary: Weighing the Benefits and Costs of Flexibility in Making Software: Toward a Contingency Theory of the Determinants of Development Process Design.” Information Systems Research, vol. 20, no. 3, 2009, pp. 462–477. JSTOR, www.jstor.org/stable/23015475. Accessed 1 Jan. 2021.

Brad A. Myers. "A Brief History of Human Computer Interaction Technology." ACM interactions. Vol. 5, no. 2, March, 1998. pp. 44-54.

Carter & Nielsen, "Using Artificial Intelligence to Augment Human Intelligence", Distill, 2017.

Coale, Kristi. “Closing OpenDoc - a Great Leap Backward?” Wired, Conde Nast, 14 Dec. 2017, www.wired.com/1997/03/closing-opendoc-a-great-leap-backward/.

Eghbal, Nadia. Working in Public: the Making and Maintenance of Open Source Software. Stripe Press, 2020.

Fayad, Mohamed & Cline, Marshall. (1996). Aspects of Software Adaptability. Commun. ACM. 39. 58-59. 10.1145/236156.236170.

Fussell, Sidney. “The Schism at the Heart of the Open-Source Movement.” The Atlantic, Atlantic Media Company, 5 Jan. 2020, www.theatlantic.com/technology/archive/2020/01/ice-contract-github-sparks-developer-protests/604339/.

Gamezpedia. “Gamezpedia/Awesome-Actionscript.” GitHub, github.com/Gamezpedia/awesome-actionscript.

Gaver, William w. “Designing for Homo Ludens.” Computer Related Design, 21 Oct. 2014.

Kopf, Dan. “R And Python Are Joining Forces, in the Crossover Event of the Year.” Quartz, Quartz, qz.com/1270139/r-and-python-are-joining-forces-in-the-most-ambitious-crossover-event-of-the-year-for-programmers/.

Hanson, Chris, and Gerald Jay Sussman. Software Design for Flexibility: How to Avoid

Programming Yourself into a Corner. The MIT Press, 2021.

Jobs, Steve, Interviewee. Memory & Imagination: New Pathways to the Library of Congress. 6 Oct. 2011, www.youtube.com/watch?v=6kaIMB8jDnY&feature=emb_title.

Moss, Frank. The Sorcerers & Their Apprentices: The Untold Story of MIT Media Lab. Crown Business, 2011.

Nick Kolakowski. “Top 10 Most Popular Open Source Projects on GitHub.” Dice Insights, 8 Nov. 2019, insights.dice.com/2019/11/08/10-popular-open-source-projects-github/.

Linnemann, Frederick G., and Carl E. Minich. “Logic Design System.” Proceedings of the June 1971 Design Automation Workshop on Design Automation - DAC '71, 1971, doi:10.1145/800158.805090.

Stark, Clancy. “Measuring the Value of Design Systems.” Figma, www.figma.com/blog/measuring-the-value-of-design-systems/.

Semuels, Alana. “Machines and AI Are Taking Over Jobs Lost to Coronavirus.” Time, Time, 6 Aug. 2020, time.com/5876604/machines-jobs-coronavirus/.

Purcell, Kristen, et al. “The Impact of Digital Tools on Student Writing and How Writing Is Taught in Schools.” Pew Research Center: Internet, Science & Tech, Pew Research Center, 30 May 2020, www.pewresearch.org/internet/2013/07/16/the-impact-of-digital-tools-on-student-writing-and-how-writing-is-taught-in-schools/.

Wessner, Charles W. “Stanford and Silicon Valley.” Best Practices in State and Regional Innovation Initiatives: Competing in the 21st Century., U.S. National Library of Medicine, 1 Jan. 1970, www.ncbi.nlm.nih.gov/books/NBK158815/.

Vanderbilt, Tom. “The Pleasure and Pain of Speed - Issue 9: Time.” Nautilus, 23 Jan. 2014, nautil.us/issue/9/time/the-pleasure-and-pain-of-speed.

Victor, Bret. “Magic Ink.” Magic Ink: Information Software and the Graphical Interface, 15 Mar. 2005, worrydream.com/MagicInk/#designing_a_design_tool.

Victoria and Albert Museum, Online Museum. “The Pioneers (1950-1970).” A History of Computer Art, Victoria and Albert Museum, Cromwell Road, South Kensington, London SW7 2RL. Telephone +44 (0)20 7942 2000. Email Vanda@Vam.ac.uk, 17 July 2013, www.vam.ac.uk/content/articles/a/computer-art-history/.

Von Schomberg, L., Blok, V. Technology in the Age of Innovation: Responsible Innovation as a New Subdomain Within the Philosophy of Technology. Philos. Technol. (2019). https://doi.org/10.1007/s13347-019-00386-3

Zuegel, Devon. “Episode 03 Ted Nelson.” Tools & Craft, Notion, www.notion.so/tools-and-craft/03-ted-nelson.