# Towards Creative Version Control

SARAH STERMAN*, University of California, Berkeley, USA
MOLLY JANE NICHOLAS*, University of California, Berkeley, USA
ERIC PAULOS, University of California, Berkeley, USA

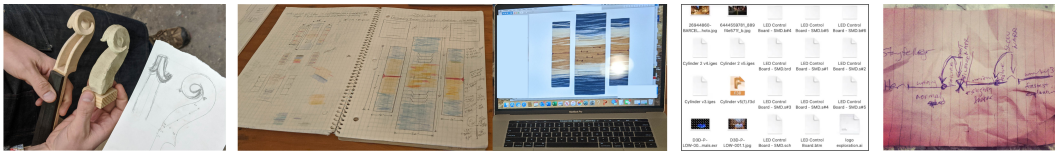

Fig. 1. Creative practitioners capture and use version histories in their creative process. A selection of *version artifacts*, from left: a Violin Maker iterates a design for a custom scroll from sketch to clay model to final carving in wood. A Tapestry Weaver captures an idea in a notebook, then photographs the final tapestry. A New Media Artist saves multiple digital copies of circuit board designs and 3D models at different stages of the process. A Physical Performer captures the rhythm of a show in a quick sketch.

Version control systems are powerful tools for managing history information and shaping personal and collaborative processes. While many complex tools exist for software engineering, and basic functionality for capturing versions is often found in collaborative applications such as text editors and design layout tools, these systems are not attuned to the needs and behaviors of creative practitioners within those domains, and fail to support creative practitioners in many others. Through 18 semi-structured interviews across diverse domains of creativity, we investigate how creative practitioners use version histories in their process. With the familiar paradigms and features of software version control as an organizing structure, we discuss how these creative practitioners embrace, challenge, and complicate uses of version histories in four ways: using versions as a palette of materials, providing confidence and freedom to explore, leveraging low-fidelity version capture, and reflecting on and reusing versions across long time scales. We discuss how the themes present across this wide range of mediums and domains can provide insight into future designs and uses of version control systems to support creative process.

CCS Concepts: • **Human-centered computing** → *Empirical studies in HCI*; • **Applied computing** → Arts and humanities.

Additional Key Words and Phrases: Documentation; Version Control; Creativity Support Tools.

**ACM Reference Format:**
Sarah Sterman, Molly Jane Nicholas, and Eric Paulos. 2022. Towards Creative Version Control. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW2, Article 336 (November 2022), 25 pages. https://doi.org/10.1145/3555756

*Both authors contributed equally to this research.

Authors' addresses: Sarah Sterman, ssterman@berkeley.edu, University of California, Berkeley, USA; Molly Jane Nicholas, molecule@berkeley.edu, University of California, Berkeley, USA; Eric Paulos, paulos@berkeley.edu, University of California, Berkeley, USA.


2573-0142/2022/11-ART336
https://doi.org/10.1145/3555756

## 1 INTRODUCTION

The need to manage prior versions of artifacts and ideas exists across many domains: writers create multiple drafts; programmers track incremental changes in large projects as they add features and fix bugs; instrument makers evolve violin designs over time. In each of these contexts, practitioners use tools to assist in managing the history of a project.

While the need for history management is common to each of these examples, such domains vary widely in other aspects – mediums, tools, outputs, traditions, and values. Yet despite these diverse needs, the structures of existing digital history management tools remain remarkably limited. Software development showcases the most widely adopted set of tools for history management in the form of version control systems (VCS). The core goals of VCS include supporting collaboration, recording changes, and reverting mistakes, in order to improve programmer effectiveness, efficiency, and collaboration [26, 42, 55]. These values recur within the design of creativity support tools more broadly: Shneiderman identifies "history-keeping" as a central design principle for creativity support tools and identifies its primary goals as recording and comparing alternatives, reverting to and modifying earlier alternatives, and communicating with colleagues [45]. These goals closely parallel those of VCS. Digital history management interfaces embedded in consumer applications – such as the timestamped lists of revert-able versions that have become ubiquitous in collaborative online tools like text editors, spreadsheets, file sharing, and design tools – commonly support these goals as well, emphasizing collaboration, precise records, reversion, and efficiency (Figure 2). Such history management tools are used by a wide variety of people across many disciplines, including creative practitioners. Yet creative practices may also have different values from those embedded in the design of software VCS. Might a visual artist prioritize a different set of values over efficiency, fidelity, or the ability to revert a 'mistake'?

Moreover, tools do not just support the goals of the users, they also shape goals and working styles [12, 28]. In software development, version control has become integral to the programming process, where capabilities like 'branching' and 'diffing' fundamentally shape how programmers structure collaboration and solve problems. As creative practitioners embrace digital history management methods, we must consider how to support history management not only as a stand-alone goal, but also as tool that shapes the creative process. Are existing capabilities of VCS equally well-matched to the working styles of practitioners in diverse domains? What are the values that best support creative practitioners, and might those considerations benefit programmers as well? Programming requires creative behaviors, especially in exploratory domains such as data science, machine learning, or creative coding, and these behaviors often do not mesh well with existing VCS [22]. While designers of software VCS have laid highly successful groundwork in history management, these tools have the potential to benefit many more users across diverse domains if they are designed with sensitivity to the needs of creative practice.

Through 18 semi-structured interviews with creative practitioners, this paper explores how past versions of work are used as materials and as tools to support the creative process across a wide range of domains. By looking across widely varying domains and mediums, we can identify commonalities in how version information can support creative process, beyond the capabilities or constraints of particular tools. Interviews covered digital practices such as software engineering, creative coding, and academic writing; physical practices such as violin making, tapestry weaving, and industrial design; and experiential practices such as physical performance and museum installations. In each of these domains, practitioners use creative processes to complete their work. These processes rely on tools to record and manipulate versions, from software tools such as git and GitHub, to ubiquitous digital data formats such as photographs, to physical mediums like paper scripts, notebooks, or cardboard templates.

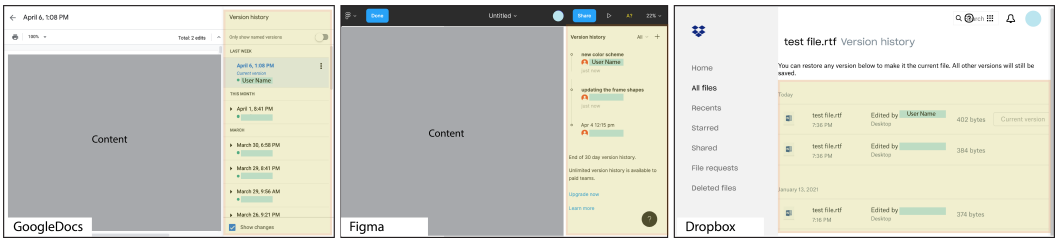


Fig. 2. Many modern collaborative interfaces such as (from left) text editors (GoogleDocs), design layout software (Figma), and filesharing (Dropbox) include version control interfaces as timestamped lists (highlighted in yellow), emphasizing precise records, reversion, efficiency, and collaboration.

With the familiar paradigms and features of software version control as an organizing structure, we discuss how these creative practitioners embrace, challenge, and complicate uses of version histories in four ways:

- While versions can represent a progression of a project from start to finish, creative practitioners also use versions as a *palette* of materials.
- The capability to rapidly revert to an old state increases efficiency and protects against production failures. Yet saving versions also provides a sense of *confidence and freedom* that encourages risk-taking and exploration regardless of the amount of effort necessary to revert to an old state.
- High-fidelity capture saves all the details of a prior state, yet the deliberate choice to leave out information and capture only in *low-fidelity* creates space for productive variation, spontaneity, and adaptation.
- Considering captured versions *across project boundaries* and on *long timescales* allows practitioners to reflect on personal process and growth, and to return to and recombine old ideas.

The ways that these practitioners have adopted, appropriated, or rejected existing version control tools reveal opportunities for better supporting the paradigms of version use in creative practice. Tools that are sensitive to the process needs of creative practitioners may be considered *creative version control systems* (CVCS). Whether adapting popular version control systems to a creative workflow, or drawing on existing history behaviors to inform the designs of new tools for creative practice, understanding the techniques in use by expert creative practitioners is key to designing CVCS that support creative process across domains.

## 2 RELATED WORK

### 2.1 History Management Tools

*History management tools* capture, organize, and support interaction with the information and artifacts that form a project history, such as documentation, commentary, specific artifacts, or versions of artifacts. Such a tool might focus on saving content, recording decisions and revisiting reasoning, or enabling group collaboration [16], while also overlapping with other purposes: a design notebook supports active ideation; a website for documenting process shapes community norms [24]; a tool for visualizing version history enhances grading and instructor feedback [54]. Digital history management tools include software version control systems such as *git*[1] or *Subversion*[2],

[1]https://git-scm.com
[2]https://subversion.apache.org

as well as tools like file sharing platforms or email, which store the history of documents or conversations. Physical examples might include design notebooks, or a filing cabinet of old drafts.

*Version control systems* (VCS) are a specific subset of history management tools that organize iterative changes to specific digital artifacts [6]. While history management tools encompass information created after the fact to explain or contextualize an artifact, version control tools focus on the artifacts themselves, with metadata created at the same time as the artifact. While the most familiar artifact type is software source code, VCS have been created for and applied to digital artifacts beyond code, such as a custom tool for tangible information design [25], or using GitHub to write books [38, 52]. In this paper, we focus primarily on history management behaviors through the lens of version control. Version control systems are particularly common and powerful tools, which are tightly bound to the creation of the artifacts themselves, and therefore integral to workflows and process. VCS are also key tools in software practices, providing a foundation to consider adaptations of existing tools to support creative process. In this paper, we broaden the common conception of VCS as applying only to digital artifacts: certain physical artifacts or tools can be fruitfully considered as versions or version control systems.

It is useful to define three additional terms as they are used in this paper:

*Artifacts* are physical or digital objects created by people. The final output of a creative process might be an artifact, such as a violin created by a luthier, or an ephemeral work, such as a performance. An ephemeral work might generate artifacts, such as an audio recording of a concert. Artifacts are also generated during the process, such as notes, tools, documentation, or drafts.

A *version* is an artifact captured at a particular point in time that is conceptually linked to prior or subsequent iterations. This is easy to imagine with digital artifacts, as they can be directly copied and modified. It also applies to physical artifacts: for example, we can understand two physical sketches as *versions* if one is an iterative change to the first. A paper sketch and a subsequent prototype might also be considered versions, despite the change in materials.

*Documentation* is an artifact or collection of artifacts specifically designed for communication about the project. This may be targeted at people other than the creators, or intended for the creators themselves in the future. In this paper, we focus on versions rather than documentation; while the two are often related, the ways they are created and used differ significantly.

### 2.2 Version Control Systems for Software

Version control systems for software development have transformed software development practices, providing essential infrastructure for collaboration on shared artifacts. Yet the conceptual models behind current software VCS have resulted in designs that do not always match the needs of practitioners. Version control, also referred to as revision control or source control, has been evolving for decades, tracing its roots back to the 1970s [40]. As version control systems grow increasingly more capable, the fundamental goals and concerns have remained relatively stable. In early systems, the focus was on identifying what changed and when, propagating fixes across versions, knowing what version a customer has, and reducing storage requirements [40]. More recent work identifies key goals as tracking reasons for changes, supporting collaboration, and allowing reversion [26], as well as coordination and organization [55]. These goals are supported by features such as merging, sandboxing, tracking history, reversion, and synchronization for collaboration [42]. These features and goals are essential to modern software development practices, and have radically improved both individual and collaborative workflows since their adoption.

Yet software version control is not always successful even among people who write code. Kery et al. show how data scientists who work with code in an exploratory manner eschew version control systems for manual strategies, like copying snippets of code [22]. These data scientists required speed, flexibility, and visibility of options for their exploratory processes, outweighing needs for

collaboration features or reversion. Similar mismatches in the values of VCS and the processes of creative practitioners are present in our findings across creative domains, emphasizing the need for alternative paradigms for version control. *git*, created in 2005, is now one of the most common VCS tools, with GitHub, a graphical, collaborative tool for working with git, reporting over 73 million developers in 2021.[3] Yet Perez De Rosso et al. note that the difficulty of learning git turns away many new users [39], and that its complex underlying conceptual model does not match how many people approach writing code. Aligning domain values with system capabilities is essential for a successful partnership between user and tool; in this paper, we explore how versioning behaviors in a broad range of creative domains both share and challenge existing values in software VCS. By understanding the ways version history is used in creative domains, we can understand how the design principles of software VCS might be adopted and adapted to better serve the needs of creative practitioners, both when working with code and with other materials.

### 2.3 Version Control in Non-Software Domains

Code is not the only material for which version control tools have been developed. For example, version control tools are common for office software, CAD, and journal articles [26]. When considering how to design VCS for CAD, Chou et al. note the importance of considering the uniqueness of the application domain, as different contexts require different capabilities [9]. We align with this philosophy as we investigate creative processes to understand the capabilities and models of version control needed in creative domains.

Despite the variation across domains, existing VCS often share conceptual models and values with traditional software VCS. Khudyakov et al. identify increasing safety and stability, and reducing conflicts or usage of incorrect versions as specific goals for VCS for CAD [23]. In text editing and office documents, supporting collaboration is again essential, with tracking history, merging, and diffing as key capabilities [11, 14, 41]. Version control is important to feedback and annotations in collaborative writing contexts, keeping comments in sync with content [53]. Zünd et al. develop VCS for collaborative story authoring in various media, including images and video, again focusing on the collaboration benefits of features like merging changes from multiple authors [56]. Klemmer et al. develop a versioning system for early-stage information design, using digital media to capture the history of a tangible interface, focusing on the capabilities of reversion, collaboration, and reflection [25]. Such designs mirror the capabilities and goals of software version control. This similarity can be both a benefit and a drawback: leveraging existing capabilities makes version control systems powerful, yet can constrain the role they play in the creative process. Of 4101 respondents to a 2020 survey about UX tools, 892 or 22% indicated that they were dissatisfied with their main version-tracking tool [36]. Shneiderman includes rich history-keeping as a key feature for creativity support tools [45], yet as we consider the role VCS plays in creative practices, we must go beyond existing models and values for VCS. To create or adapt VCS effectively for creative domains, we must understand how practitioners use version information to shape their own process, engaging how different materials and workflows affect history behaviors. In this paper, we identify four themes that describe creative practitioners' uses of version histories, to broaden our understanding of how VCS can support creative domains.

### 2.4 Version Control and Creative Process

In this paper, we are interested in understanding how tools for version control inform and support the creative process. Tools, including version control systems [25], are not just things to be used, but influence the process and the user in return [12, 28]. To ground our approach to studying

[3]https://github.com/about; retrieved Nov 1 2021

creativity, we draw on Amabile's Componential Model of creativity, in which there are three core aspects: domain-relevant skills, creativity-relevant processes, and task motivation [2]. Here we focus on the second component, creativity-relevant processes [1]. Kaufman and Beghetto identify particular levels of creative practice [21]: our interviews focus on professionals, the "Pro-c" level, with significant experience and established success in their fields.

Many domains and practices are creative, even if they are not colloquially considered creative the way that art and performance are. We take a broad view of what domains are creative, as an area in which the practitioner utilizes creative process. For example, software development is creative, as it requires open-ended problem solving and the creation of contextually novel solutions [31]. There is no single "correct" process among programmers, and programming process has parallels in other creative disciplines [50].

Frich et al. explore how creative practitioners use digital tools in their process across five domains of creative practice [15]. We align with the value of exploring multiple, diverse domains to gain insight into commonalities of creative process for the benefit of digital tool design. Li et al. interview visual artists to understand how they use and create software tools in their artistic practice [29]. We use a similar method of in-depth interviews to understand creative practice, with a focus on versioning behaviors across domains. Li et al. discuss how mismatched values between the practices of visual artists and software developers reduce the adoption and usefulness of existing software tools to visual artists; similarly, we find that a mismatch in values between existing version control systems and versioning behaviors in creative process limit the adoption and usefulness of VCS for creative practitioners.

Large-scale version control systems are not the only way to think about process interactions with history data. "Undo," for example, is a ubiquitous feature in computational tools, allowing the reversion of mistakes on a small scale. The ability to undo is important to creative process to make temporally proximal changes, for example as explored in painting by Myers et al. [33] and image manipulation by Terry et al. [49]. Myers et al. additionally investigate how to support a "natural" approach to exploratory coding, integrating more complex backtracking in a code editor without requiring explicit version control [34]. Terry et al. discuss the importance of *variation* and *experimentation* to creative practitioners, exposing how creative practitioners appropriate the capabilities of existing software to store proximal history alternatives, such as using layers in photo editing software to store versions within a single file. They focus on near-term history behaviors to support reflection-in-action [44]. Jalal et al. explore the importance of version histories for choosing color palettes, and integrate versioning into color pickers, which are usually a component of a larger system [20]. In this paper, we investigate the process-focused uses of history information to identify high-level themes that cross domain and tool boundaries. We discuss some similar values, such as the importance of alternatives and re-use of histories, expanding the context for these behaviors to a wider set of domains and longer time periods.

We present the idea of creative version control systems (CVCS): version control systems designed to support the process needs of creative practitioners. To motivate this approach, we use the lens of creative process to examine two existing CSTs that address version histories. First, Variolite is a versioning tool for "exploratory programmers" such as data scientists [22]. Exploratory programming is a creative domain, requiring open-ended problem solving and creative exploration. Existing VCS do not support data scientists in their needs for fast interaction, quick comparisons of options, and versioning of small components; instead, data scientists used informal versioning practices such as copy-pasting from other files. Kery et al. identified the process needs of data scientists, and foregrounded those paradigms in the design of Variolite, while also providing the benefits of a formal versioning tool. By discussing a tool like Variolite through the frame of a CVCS, we gain a generalized way to address the importance understanding process and elevating the

values of a creative practice in the design of version control tools. A second example, Knotation, is a documentation CST for choreographers that incorporates basic versioning [10]. This tool draws from particular needs of choreographic practice in its design of information representation and exploratory features. Knotation supports version control for the ability to revert to previous states. A CVCS design lens might enrich the possibilities created by Knotation by considering other ways version histories might be valuable to choreographers' process: Might choreographers using a digital tool like Knotation benefit from a *palette* mindset? Would *low-fidelity* representations enhance the choreographers' reported desire for "informality" and "imprecision"?

## 3 METHODS

### 3.1 Participants

To explore and understand creative process, we interviewed 18 expert creative practitioners across a wide variety of disciplines (Table 1). Participants were selected from domains that require novelty and open-ended problem solving, where practitioners must use creativity skills in daily work [21]. We began by selecting sites and interviewees according to an *a priori* set of distinctions that seemed most likely to generate a broad range of behaviors: physical, digital, or ephemeral mediums; extent of collaboration; and use of computation in daily work. We chose subsequent creative practices to maximize the range and diversity of experiences as our understanding evolved, in concert with our research questions. Following Charmaz's Grounded Theory approach, we chose additional practices within this frame that would support theory construction, rather than seeking a representative population across "all" creative practices [8].

Each participant is an expert in their field, with a mean of 18 years of experience (range 5 to 47 years). Among the 18 participants, there were 6 women and 12 men. Interviews took place either at the participants' primary workspaces (8 interviews), to enable better understanding of their tool use in context [4, 47], in a public location[4] (1 interview), or over video conferencing software (9 interviews). During the COVID-19 pandemic, all interviews were held remotely over video conferencing software for health and safety. During video interviews, participants used screensharing and their webcams to show their tools, workspaces, and creative outputs.

### 3.2 Interview Methods

Interviews were semi-structured, guided by grounding themes of artifact use and collection of information over time, and shaped by the individuals' practice and reflections. Each interview lasted between 1 and 2 hours, during which we asked a semi-structured set of interview questions, focusing on their personal creative practice and background. To ground our discussion in concrete examples of daily work [4], the topics centered on how each practitioner creates artifacts and versions, how artifacts and versions are used in their process, the tools and materials they use, for how long artifacts and versions are kept and why, and the roles artifacts and versions play in the creative process. Following best practice for Grounded Theory [7, 8], we evolved our research questions as we went, focusing more specifically on version control tools and frameworks in later interviews. Participants were asked to walk through concrete examples of their workflows, which served as a starting point for surfacing details about their personal working style.

Using a recent project of the participant as a grounding example, each participant was asked questions such as *How do you record versions of an ongoing project? What forms of documentation do you use in your creative process? How do you assess your growth as an artist over time? Do you revisit past artifacts/histories from old projects? How do you explore alternatives?* and *What tools do you use*

[4]The Physical Performer rents workspaces temporarily in various cities due to work travel. The interview was held in a public location; the participant showed pictures from prior rehearsal spaces and materials.

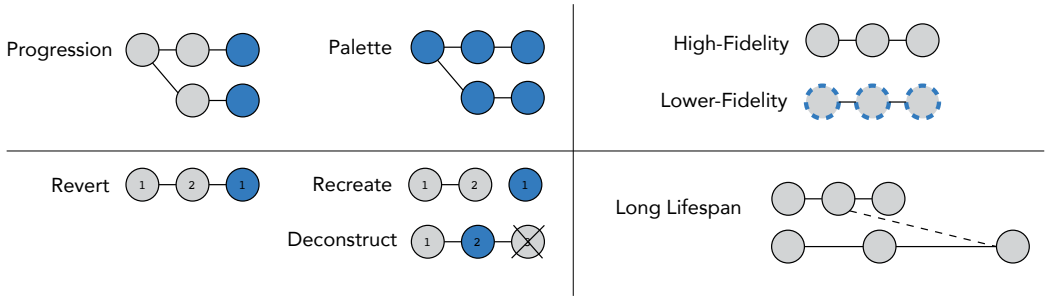


Fig. 3. Four structural paradigms of standard version control are embraced, challenged, and complicated by creative practitioners: approaching versions as a *progression* towards a goal with only the most recent versions active (indicated by blue nodes), or as a *palette* of options, where all versions are concurrently active; gaining confidence and freedom through the ability go back to earlier states (indicated by blue nodes), whether through *reversion*, *deconstruction*, or *recreation*; choosing *high or low fidelity representations* of past versions to create space for variation in future iterations; and creating and revisiting version histories over *long time periods and across projects*.

*during different stages of your process?* Our interviews are interactional events [48], in which the questions evolve in response to participant background, shaped by earlier interviews. Following best practices for Charmaz's Grounded Theory [7], we simultaneously engaged in analysis and data collection, iteratively constructing our analytic frame and updating our question prompts for future interviews as we synthesized and identified emerging themes. We read and analyzed all interview data and discussed all emerging themes [32].

### 3.3 Analysis

We analyzed the interview data using Grounded Theory. Grounded Theory has three main branches: Strauss and Corbin; Glaser; and Charmaz [43]. We embrace Charmaz's approach, a reflexive research style in which knowledge is co-constructed between interviewee and researcher [7, 8]. Our analysis is interpretivist, rooted in the social construction of knowledge and polysemic understandings of truth [27]. To perform thematic analysis, we first transcribed each semi-structured interview, then performed open-coding [46] on the transcripts. We iteratively reviewed and refined these into a closed set of codes, which we then re-applied to the transcripts as we performed additional interviews. We read and analyzed all interview data and discussed all emerging themes [32]. Themes relating to version control tools are presented below. We additionally uncovered themes relating to creative process, creative cognition, motivation, and emotional affect; for a full discussion of these topics, see [35].

## 4 FINDINGS

In the analysis of the interview data, we identified four structural paradigms of standard version control that are embraced, challenged, and complicated by creative practitioners (Figure 3). In the following sections, we first introduce the range of mediums and tools practitioners used to capture and interact with version history, then present each paradigm through a selection of versioning behaviors participants used in the creative process to inspire, explore, create, and reflect.

### 4.1 Creating Version Histories with Diverse Materials and Tools

Version histories are created and captured with a wide variety of tools and mediums. By discussing and comparing different approaches, we can learn from a wide range of creative techniques and behaviors. Here, we introduce a few of the types of tools and mediums that arose in our interviews, to ground the following discussion of creative behaviors that rely on these versioning tools.

*4.1.1 VCS for Software.* Among creative practitioners working with code, some participants used established, commercial version control systems such as git or company-specific version control systems, and tools such as GitHub (Software Engineer 3, Software Engineer 1, Generative Artist). However we also saw some practitioners who have used these tools professionally nevertheless eschewing them in their creative work, either *manually* saving new copies of a file when making changes, or building personal, *custom* tools (New Media Artist, Creative Coder). For example, while the Generative Artist does use git in his creative process, he adapted the existing interaction paradigms of git to his personal creative workflow by building a *custom* toolchain.

*4.1.2 Manual Versioning.* Manual versioning, where the practitioner saves new copies of digital files to track version changes, was used by the New Media artist for code, as well as by the Animal Behavior Researcher for text documents and spreadsheets:

> *Animal Behavior Researcher*: I am that person that has 8 million different versions of every Word document. It'll say like "use me" or "no, use me" or "final final final", or "final final final final version."

Both the Animal Behavior Researcher and the New Media Artist encountered challenges with the manual approach, where errors were more easily introduced into the workflow and key information was lost or forgotten through uncertainty about which files contained which changes, which version a collaborator was working with, or why certain copies had been made.

*4.1.3 Digital Capture.* Physical and ephemeral outputs were sometimes captured digitally. Digital mediums such as audio and video were used by performers to capture versions of ephemeral performances as they developed them during rehearsals (Physical Performer, Director). Photographs were essential for the Industrial Designer to capture intermediate states of physical prototypes.

*4.1.4 Physical Capture.* Paper was a common physical medium to capture versions in a physical format. The Performer used poster-sized scripts to share version state with a collaborator. Sketches and notebooks were common across a variety of physical practices (Tapestry Weaver, Industrial Designer, Violin Maker, New Media Artist) but not limited to physical practices, as practitioners who worked mostly in the digital world also used paper sketches, notebooks, and print-outs to capture early versions or create long-term archives of later versions (Animal Behavior Researcher, Academic, Software Engineer 3).

Version information was also captured in physical materials beyond paper. Physical prototypes encoded version information for the Industrial Designer. The Violin Maker retained version histories of his instrument designs in the templates and molds he used to carve and shape the wood; a new design requires an updated set of templates.

Whether the versions were captured digitally or physically, from originally digital, physical, or ephemeral works, the version data was essential to the creative process. In the next sections, we discuss four ways version histories support creative process.

### 4.2 Using Versions as a Palette of Materials

Version control systems often represent history as a sequence of serial versions. The most recent version represents the active, correct state, in a linear progression towards an improved outcome.

| Participant | Primary Output Medium | Digital Versions | Physical Versions |
|---|---|---|---|
| Academic | Digital | Text documents | Handwritten notes |
| Animal Behavior Researcher | Digital | Text documents, spreadsheets | Handwritten notes |
| AR/VR Artist | Digital | Text documents, screenshots, slideshows | - |
| Creative Coder | Digital | Code files, images, P5LIVE | - |
| Design Lead | Digital | Slideshows, photographs, wireframes | - |
| Generative Artist | Digital | git commits, images | - |
| New Media Artist | Digital | Code files, 3D models, P5LIVE, circuit schematics, Adobe Photoshop layers | Breadboard circuits, prototypes |
| Software Engineer 1 | Digital | git commits | - |
| Software Engineer 2 | Digital | git commits | - |
| Software Engineer 3 | Digital | git commits | Printed out code, whiteboard notes |
| Museum Curator | Experience | Photographs | Handwritten notebook; binder with print-outs, hardcopies, and notes |
| Personal Stylist | Experience | Slideshows, photographs | Moodboards |
| Director | Performance | Video recordings, photographs | Photographs, handwritten notebook |
| Physical Performer | Performance | Audio recordings, photographs | Butcher paper |
| Ceramicist | Physical | Photographs | - |
| Industrial Designer | Physical | Photographs | Prototypes, sketches |
| Tapestry Weaver | Physical | Photographs | Sketches, handwritten notes, swatches, weavings |
| Violin Maker | Physical | - | Handwritten notes, paper templates, wooden molds, clay models |

Table 1. Selection of mediums and tools used by participants to capture version histories in digital and physical forms. Many practitioners use both digital and physical versions.

While the concept of 'branching' allows exploration of alternative paths, there is commonly a single main branch that is considered the true current state of the project.

Software Engineer 1 is a software engineer at a large technology company, who has worked professionally with code for ten years. He describes how he uses version control within his company:

> *Software Engineer 1*: We have the true copy of all the code, and then people can make deviations of that and then resubmit them back to the true copy.

This approach to development is a highly productive process technique when the creator or the team is pursuing a single, known goal. However, for many creative practitioners, the paradigm of a linear progression towards a single measurably better result is at odds with their process. Instead of considering version histories as a document of the past, they use versions as a palette of resources to enable a conversation with materials. All three practitioners who worked in creative code (New Media Artist, Generative Artist, and Creative Coder) use this approach, as well as the Tapestry Weaver and Animal Behavior Researcher.

The New Media Artist is a creative coder who has been working professionally on digital and hybrid artworks for twelve years, and teaching digital and electronic art for seven. The New Media Artist approaches code as a material, much in the same way a physical artist might work with paint or clay. In contrast to the "top-down" approach of working towards a goal, he calls this a "bottom-up" approach:

> *New Media Artist*: I don't have an end goal at first...it's more or less how artists usually start their practice, they play with sculpture, they play with clay, they mold it and then they look at it and along the way [they say] "Oh, this is the direction that I want" ...so [when coding] I come up with "Oh, I want to create an easing function," or "I want to move one point to another point and leave a trace." And once I implement that ...they become my modules – materials – to apply to different sketches.

In this approach, versions of the modules are not progressions towards a goal, but variations on a material, acting as a palette of paints or a selection of brushes. The New Media Artist saves all these versions as separate files, so that he can access them in parallel and quickly swap between alternatives.

Though the New Media Artist is experienced with version control, and has used such systems professionally as a developer, he does not use a version control system in his personal creative process. He shares his feeling that the mindset of progression is counterproductive to the artistic process:

> *New Media Artist*: I think the mentality of *git* is: there's only one version that's keeping progress, but in art, progress doesn't really mean something. I want diversity, I want different versions, not one final best version.

Versions are essential to the New Media Artist's process, but his use of versions is at odds with the standard model of progression inherent to the design of tools like git.

The Generative Artist programmed professionally for seven years before discovering creative coding; he has been creating generative art with code for a year and a half. He is also an oil painter, working in traditional media for seven years. In his code-based creative work, his approach to version control is very different from his professional programming work: like the New Media Artist, past versions are a palette of options rather than a progression towards an end point.

Each time the Generative Artist modifies his code, he generates dozens of image outputs and saves each one. These outputs offer a set of options from which he can choose the most interesting or inspiring direction to continue pursuing. Often he cycles back to earlier versions to explore new directions or find new inspiration, regardless of how long ago they were created.

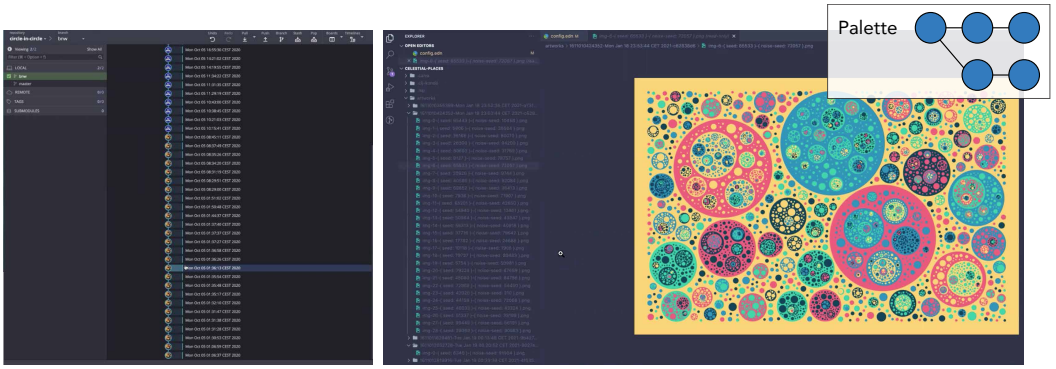


Fig. 4. The Generative Artist creates hundreds of commits that capture the complete state of each generated output (left). Though displayed as a chronological history, the Generative Artist uses these commits not as a linear history of improvement, but as a palette of options that he cycles back through to find new inspiration and pursue new directions. He navigates these commits by the associated images, saved in a separate folder (right).

*Generative Artist*: I sometimes go back and look at my old art: maybe [I] can just do something with this, maybe mix it with different colors and see how things pop up.

We see this behavior of keeping past work accessible as a palette of options in a physical domain as well: the Tapestry Weaver deliberately sets up her studio to make past outputs accessible for inspiration or reworking. The Tapestry Weaver is a fiber artist, and has been weaving for 43 years. Her tapestries are handmade on looms in her home studio, each one an effort of weeks or months (Figure 7). She hangs many of her pieces on her walls, especially ones that were "unsuccessful." She keeps these pieces visible and available so that she can "rework" and "play with" them in the future. Sometimes this will be as inspiration to a new piece, or a direct modification to the old piece. She keeps notebooks and sketches of designs, along with photographs of the final outputs; sometimes she returns to an older design to weave it again with new colors or techniques. Like the creative coders, each version remains available as a palette of inspiration and a material to become a new design.

The Animal Behavior Researcher has been working in veterinary research for eleven years. Even in scientific domains, writing often has strong connections to creative process. She often has to write grant proposals, journal articles, and presentations, where she uses a hybrid approach to versions: while she keeps a single "current" copy of each document, she manually saves all past versions in an accessible folder. Like the New Media Artist, she values these easily accessible alternate versions, using them not as linear historical records, but as a selection of materials she can repurpose and recombine.

*Animal Behavior Researcher*: [For] a grant, I need this snippet here and that snippet there so there is a lot of...reusing or recycling or adjusting a lot of things you've written in the past.

A journal may want one version of a writeup, and a grant another. Neither is necessarily more correct or complete than the other; they exist in parallel as materials for reuse. For the Animal Behavior Researcher, she needs access to these versions "without necessarily rewind[ing] or undoing things."

While manual versioning does allow parallel interactions, it is also an error-prone method: which version contains which changes, which version a collaborator is working from, or even why a version was made can be unclear. The New Media Artist and Animal Behavior Researcher both run

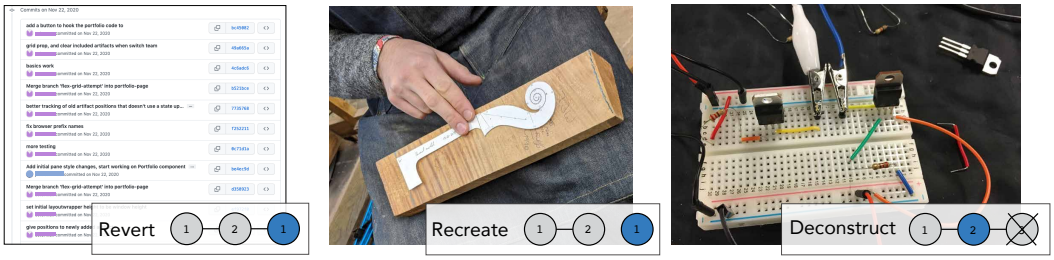


Fig. 5. Practitioners across domains store old versions to provide confidence to explore new alternatives. From left: a git history provides rapid, low-cost reversion to old states by reloading a prior commit. Physical objects require more labor: the Violin Maker must carve a violin neck from new material to return to an old state; breadboarded prototypes can be deconstructed to return to an old version captured by a photograph.

into these problems. The Generative Artist, in contrast, has adapted git to serve his process as a palette, using custom scripts and additional software. Each time he generates an image, his custom scripts automatically save the image file and auto-commit the code. The image filename contains all the configuration variables as well as the commit hash, and the commit message contains the filename of the generated image (Figure 4). This allows him to match each output with the version of the code that created it, complete with all random variables, and therefore allows him to recreate any image at any time in the future. Yet his interactions with the images are entirely separate from git, and rely on image curation tools. He can work around the chronological presentation of commits, but it provides no significant benefit to his workflow.

From creative coding to research to tapestry weaving, the framework of versions as a palette of materials supports bottom-up, material-centric approaches to the creative process, as well as enabling inspiration, re-use, and re-combination of past ideas. Yet standard VCS tools do not natively and effectively support this mindset.

### 4.3 Gaining Confidence and Freedom to Explore through Reversion, Deconstruction, or Recreation

Reversion, going back to an earlier state of the project and continuing forward from that point, is a highly valued capability of version control systems. Practically, it allows the easy undoing of mistakes, increasing programming efficiency and the uptime of production systems. For creative practitioners across a wide range of domains, the key benefit of reversion is psychological: a sense of confidence, safety, and comfort that enables radical exploration and risk-taking. For some practitioners, reversion need not even be easy; capturing version information is sufficient to gain the emotional benefits even if returning to the earlier state would require significant labor.

Software Engineer 3, who has been working as an engineer for 5 years in Research and Development for a wireless technology company, described the benefit of reversion to his process:

> *Software Engineer 3*: [Committing] is kind of an insurance policy. Because a lot of times I'll make a change and I'll break something, and then I won't remember how I got there. So any time something kind of works, or I feel like I hit a milestone or a checkpoint, I'll make a commit so that I know I can at least get back to that point.

Because he can revert a commit to an earlier, working version, he feels free to make potentially breaking changes and explore solutions without fear. Similarly, the Generative Artist uses commits to allow him to return to any prior version, as his aesthetic intuition desires. Though he does not

conceptualize alternatives as "correct" or "working" in the same way as Software Engineer 3, the feeling of support for exploration is the same.

> *Generative Artist*: I wasn't exploring [in oil paints] and this medium [creative coding] allows me to explore, because now, I know I can just undo and get back to a state, so I do not have the fear of "Oh, I did something nice I don't want to lose it".

The New Media Artist also values the same feeling of safety, but does not use a formal version control system. Rather he saves duplicate copies of files before making a big change:

> *New Media Artist*: Having a file saved in there before I made the change, made me feel I can always turn back. I just feel safe.

Using history as an insurance policy to enable experimentation also showed up in history management behaviors in domains that work in ephemeral performance and in physical materials. The Physical Performer sought such freedom to take chances when developing a physical comedy show. As the Physical Performer and her collaborator improvised together to design their show, they captured their evolving ideas in a scribbled "script" on butcher paper:

> *Performer*: [On] a huge poster-size paper... we would write down "[A] grabs napkin, [B] double-takes, [A] this," every little minute movement.

Recording the "choreography" was essential to supporting free improvisation and exploration. Capturing even an extremely simplified form let them play, and freely negotiate about the show:

> *Performer*: [The butcher paper] was the space where we agreed on what was going to happen, and so if we were ever playing and someone did something else, [and] then the other person was like "wait I don't know", we could refer back to [the poster] and be like "is this the best way, or should we do how we just improvised and change this thing". And then often times we *would* change it, but it helped us continue to anchor back to something.

The Physical Performer understood the butcher paper as an "anchor" to their initial creative idea. Capturing concepts allowed the collaborators to experiment freely without fear of losing access to their original creative intuition, or of forgetting something that had worked better. In this way, the butcher paper script is a tangible version history, providing the same feeling of safety while exploring that Software Engineer 3 gains from his version control software.

Exploration was also core to the Industrial Designer's process, and like Software Engineer 3 and the Physical Performer, he gives himself freedom and confidence to explore by capturing version artifacts. However, the amount of labor required to return to a prior state complicates the idea of reversion for physical materials. The Industrial Designer has worked in many domains across his 23 year career, including automotive design, toys, medical devices, restaurant and museum experience design, and consumer electronics. He also teaches design, prototyping and sketching and runs a makerspace. Before he makes significant changes to a physical prototype, the Industrial Designer takes photographs of the current state:

> *Industrial Designer*: I documented it so I'd have a recording of it, and if all else is ruined I still have the recording of it. I'm allowed to take chances.

However, the Industrial Designer must destructively undo changes to the physical prototype or rebuild a new one to return to the old state. One cannot automatically revert a physical prototype from a photograph. Despite the additional labor required by the medium, the photograph is still sufficient to provide the sense of safety that is necessary to support exploration.

In digital version control contexts, whether supported by a VCS or by manually copying files, the digital representation allows reversion with minimal labor from the user. A single command is typically enough to automatically recreate the state of the system at the previous point in time. In the cases of the Performer and the Industrial Designer, the version information is enough to recreate

the former state, but requires labor by the user. The Industrial Designer must destructively undo changes to the physical prototype or build a new one to return to the old state, either *deconstructing* or *recreating* to reach the prior state. Likewise, the Violin Maker must carve a new violin to try a new direction; he cannot revert subtractive carving operations on an instrument, and so must create a new one. The Performer can discuss and remember a version by referencing the script, however she and her collaborator must re-enact — *recreate* — the scene to truly return to the prior state. Yet deconstruction and recreation provide the same benefits to these practitioners as reversion does to programmers: confidence to explore. In these cases, the amount of labor required to return to the prior state is less important than the knowledge that the version information is saved, and could be returned to if necessary.

### 4.4 Opportunities for Variation through Low-fidelity Capture

In software version control systems such as git, the information stored for each version represents a complete copy of the code content at that particular moment in time. Such a representation is *high-fidelity*, containing all the detail of the system state needed to recreate that content exactly as it was. However, practitioners did not universally value capturing complete detail. The amount of information stored at various points during the process varied widely, from complete snapshots of the entire system to the briefest of summaries. The choice of how much detail to capture was deliberate, in order to support productive variation, spontaneity, and adaptation.

For example, the Physical Performer deliberately omits detail in her captured versions in order to maintain a sense of spontaneity and liveness in her performances. The Physical Performer has been working in performance for 22 years and is trained in mime, acrobatics, and physical comedy. She creates, directs, and performs one-woman physical comedy shows. While developing a spoken comedy show, the Physical Performer iteratively developed her content over 3-4 weeks by improvising from notes she had taken about moments in her life. She performed in front of a workshop audience, improvising her movements and stories as she went. These performances were early iterations of her show, from which she would later select good parts, abandon bad parts, and rearrange the content into a full-size show.

She recorded these performances for later review; these recordings act as version artifacts for the show under development. But rather than recording video, which would capture all the details of both sound and visuals, she only recorded audio:

> *Performer*: Someone told me: "you should be videotaping this because...all your movement[s] are part of it." I never videotaped it... I need to keep some aspect of it unplanned so that I have this feeling of spontaneousness.

Capturing a version artifact is important to her process: using the audio recordings, she can remember particular phrasings that worked well, select individual parts, and recombine her stories in a later version. Equally important is *not* capturing the video: in this way, the performer allows the movements to develop and retains liveness in future iterations, rather than feeling scripted and restricted. In the context of a physical show, where visuals and audio are both essential material components, the audio-recordings are a form of low-fidelity versioning that supports a creative process that maintains liveness in future iterations.

We can also see the effects of using video to capture an ephemeral art form through the experience of the Performance Director, who has worked as an acrobat, clown, producer, director, and playwright for shows ranging from theater to circus for the past 47 years. In addition, he is an accomplished juggler. When watching a juggler in person, one cannot catch all the details of a trick. These errors can be productive, enabling the trick to evolve:

> *Director*: Those little errors [are] like a little genetic mutation, generation to generation.


Fig. 6. The Violin Maker captures version histories in a variety of physical forms. From Left: The Violin Maker's workshop. Gradation diagrams used to record the depths of the top and back of a violin. Demonstrating a paper template for carving a neck and scroll on the partly-assembled instrument. The Violin Maker's notebook in which he records designs and modifications, showing versions of the neck template alongside the final template.

These mutations contribute to each juggler's unique style, and to the evolution of juggling as a field. These days, videos of juggling techniques are easily accessible on the Internet, and able to be replayed over and over to tease out the details:

> *Director*: I don't think we lose [the mutations], I think that still happens with video. [But] it doesn't spread as fast.

Video slows the process of evolution by reducing the space for serendipitous variation. In this example we see a case where the amount of detail and accuracy of the past version, captured in memory or in video, changes how an individual's style develops.

Low-fidelity version artifacts can also support adaptation to material requirements. The Violin Maker works in the Cremonese tradition of instrument making, carving each instrument by hand from an ever-evolving set of molds and templates and flexibly adapting to mistakes and variation. He has been a professional luthier for 18 years, creating new violins and repairing old ones in his studio. Instrument-making is a deeply creative practice; the luthier is both artist and artisan as he explores new aesthetic forms and works intimately with new materials and tools to create unique instruments.

A new instrument design is developed in concert with the creation of the instrument itself: sketches become templates that are used to carve rough shapes; the depth of material across the back and top of the instrument are recorded on gradation diagrams after carving (Figure 6). The Violin Maker tracks these dimensions, shapes, and templates in order to maintain a history of successful and unsuccessful approaches, and to scaffold experimentation with new designs. However, each piece of wood has its own character, and requires unique variation on the recorded designs. Here he describes the character of a blank that will become a violin back:

> *Violin Maker*: I know that it's going to be a little soft towards the outside, because the grain should be straight as possible, and this particular piece of wood, it's slanted like that... So I know that towards the end, towards the edges, I'm going to have to make it a little thicker. So I'll go to the violin that has the same kind of density of wood, and take those thicknesses, make it a little thicker, and take it from there. Start there and see where it ends. At the end it's just feeling.

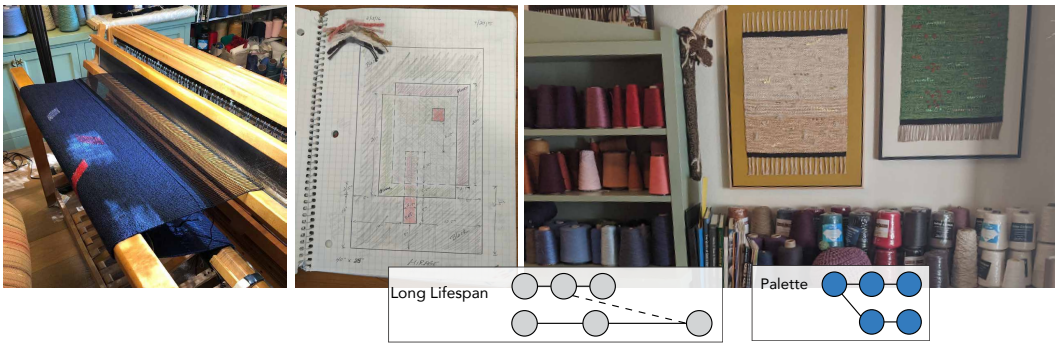


Fig. 7. The Tapestry Weaver's studio contains in-progress work on the floor loom (left), notebooks of designs (middle), as well as completed pieces hung on the walls (right). These completed pieces act as a palette of inspiration and options for re-work, if she is dissatisfied with the result. Designing, making, and re-using pieces can occur over years or decades, resulting in long lifetimes of use.

The Violin Maker engages in a conversation with his materials [44] as he works with a particular piece of wood. The collected history of his designs – versions of templates, gradation diagrams, and other notes – allows him to build off of earlier knowledge without starting over. The space left by the low-fidelity representations makes room to adapt to the needs of a particular piece of material. This space for creative variation is similar to that created by the Physical Performer when she excludes video capture of her performances.

Both low-fidelity and high-fidelity representations of past versions have their place in creative process. Low-fidelity version capture supports the techniques of creative process discussed above: the Performer creates space for spontaneity; the Director embraces productive error; the Violin-Maker adapts to materials. A familiar example of a process technique supported by high-fidelity capture is that of code reversion, where software version control systems use exact representations of earlier states to allow the programmer to return to that state. Though both ends of the spectrum support certain techniques, the wrong level of fidelity can also stymie other techniques. Therefore the default approach of both capturing and presenting all detail for past versions may not be appropriate at all points during the creative process. Selectively choosing what to capture, or capturing all data but selectively choosing what to present to the user, may allow version control systems to support a wider variety of process behaviors.

### 4.5 Using Versions across Long Lifetimes

Inspiration and iteration may influence work across years or decades; a project though complete may resurface again to be continued, revised, or dramatically altered. Creative process cannot always be cleanly divided into individual projects. For several of our interviewees, history information and project versions had influence on the creative process far beyond the lifetime of the project itself.

For the Tapestry Weaver, reflecting on her past work revealed how long certain themes had been incubating in her work, and gave her inspiration for direction for further evolution. For example, in her recent work, she has been playing with treating her weaving as a "canvas" onto which she sews other pieces of fabric. However she realized this is a much older idea than she had believed, when reflecting on prior pieces that she had kept available in her studio or had documented through photographs:

*Weaver*: [I had thought] that it is only in the past two or three years that I have been [doing] what I [call] 'weaving a canvas' and then stitching some things on top of it. And yet I did it there [on an

older piece]. And I did it there. ...And that would be decades... later after it had stewed around for a while. Which is one of the reasons why I do like to have some of my things around me, because they continue to inform what I might want to do.

In addition to reflection, the Tapestry Weaver sometimes reworks projects from years before that were unfinished or unsuccessful, turning them into new or modified pieces. Similarly, all three creative coders looked back at prior projects and version histories, either to find and reuse specific components in a current project, or to inspire new pieces of art. These version histories are equally relevant to new projects as they are to the one they were created for, and retain their relevance even over long periods of time:

*Generative Artist*: I would never push my company's production code from a year ago, but I might want to go back to an artwork that I did a year ago [to] explore it further.

In these cases, the long time frame can be a benefit in itself, allowing skills, techniques, and ideas to develop. The AR/VR Artist, Generative Artist, and Weaver all valued this growth over time, and valued the way their artifacts embodied these personal changes. As the Generative Artist described:

*Generative Artist*: As I'm maturing more and more, I can look back at my artwork, critique them but also... just have a better understanding [that] I can push them into this new direction, or maybe I can combine my new artwork [with] something from the old, and create something else new.

Reflecting on prior versions over time can reveal changes in the creator's understandings, concepts, or interests. For example, the Academic uses his old notes and free-writes to understand his evolving thought process. The Academic is an advanced graduate student in a technical field at a university in the United States, who has thoughtfully crafted his tools and habits to support his research process. Reviewing old notes not only reveals his growing understanding, but also encourages him by documenting his improvements:

*Academic*: A lot of research is coming up with the framing of an idea, about what makes it valuable, how it fits into the state of the art... Seeing that framing slowly change over time is helpful, both for recovering from false starts, and also to see that progress has been made in an otherwise very low feedback, very discouraging field of work.

However, version control systems are often focused on the project lifespan, supporting process behaviors within the creation, maintenance, and sunsetting of the project itself, but not intended for cross-project behaviors.

Software Engineer 3 discussed the same values that we see held by the Academic and Weaver when considering the potential of reflecting on old code. However, he did not engage in this kind of reflective review of old versions:

*Software Engineer 3*: I think my memory of how I solved problems before can inform my future decisions, but I don't reflect on my old code. I would like to, to some extent, but ...I'm not coming into contact with code I've written three, four years ago.

While VCS is often used to increase efficiency and productivity, it also has the potential to support the long-term development of the practitioner through reflection. Visualizations of version control data are one path towards supporting reflection, such as how the classroom tool Pensieve provides insight to students and instructors on individuals' approaches to writing code [54]. Resurfacing specific entries in a reflective context, as we see the Academic and Weaver doing, may allow practitioners to see how they have grown in specific areas, or inspire them to return to certain themes, especially across larger time scales. Version control is uniquely situated to support such reflection, as it is already a repository of rich information about past process and content.

| Theme | Illustrative Design Recommendations |
|---|---|
| Palette | - Provide rapid, parallel access to different versions without requiring a full reset of the workplace state.<br>- Outputs should be visually accessible and directly linked to the version state. |
| Freedom | - Deprioritize rapid reversion in favor of supporting confidence and freedom to explore. |
| Fidelity | - Support variation, spontaneity, and adaptation.<br>- Dynamically change representation to fit the needs of different stages of the creative process. |
| Timescale | - Make version data accessible and visible across longer timescales and multiple projects to support personal reflection and reworking of ideas. |

Table 2. Themes and illustrative design recommendations for creative version control systems. These are neither a complete nor required set of guidelines, but were commonly used and needed among our interviewees.

## 5 DISCUSSION: ADAPTING THE PARADIGMS OF VERSION CONTROL FOR CREATIVE PROCESS

Through the interviews with creative practitioners, we have seen myriad ways that version artifacts and history information support the creative process. Existing VCS features can be powerful tools for particular process techniques, and at the same time, they can be limiting for others. Here we discuss how these results can inform design decisions for version control systems, and propose Creative Version Control Systems (CVCS) as tools that foreground the roles version control systems play in creative process.

### 5.1 Creative Version Control: Supporting Creative Process by Modifying VCS

The mindsets and requirements of creative practice differ from the standard models and features of version control systems. Therefore, we cannot adopt existing paradigms of version control wholesale into new creative domains, or expect them to fully support creative process behaviors in domains that already use VCS. Instead, Creative Version Control Systems (CVCS) should be designed with supporting the needs of creative process as a central goal.

The Violin Maker discusses a compelling example of the failure case of adopting prior mindsets directly into a new practice: incorporating CNC machines into the violin making process. A CNC tool integrated into a digital version control system for a violin maker could use high-fidelity 3D scans to capture version information, and automatically return to old design versions by CNC carving new parts, with little labor from the artisan.

Yet to the Violin Maker, this is the wrong way to integrate a CNC tool into his process. Scanning a violin to capture a high-fidelity version of the design, then using a CNC to revert to that state results in poor violins that cannot adapt to the needs of the wood or accommodate creative alteration. Since each violin must be made from unique material, each instrument requires its own touch:

> *Violin Maker*: [You] have to have a method that's flexible, and that you can adapt to every piece of wood...What I want to do is have this method where the machine cuts just enough, what [an apprentice] would do for me.

Instead of reproducing the capabilities and values of standard version control for precise capture and easy reversion, the Violin Maker would prefer to leave space for adaptation to the individual piece of wood, and creative variation in the design by reducing how much of the design the CNC

cuts, and the level of fidelity captured by the version data. This approach to the integration of a CNC still imports the benefits of rapid manufacturing and offloading repetitive labor, while remaining sensitive to the needs of his creative process. By foregrounding the process needs of the creative practitioner, this more desirable design would be a CVCS.

The four themes discussed in Section 4: Findings represent a set of design choices for a CVCS that can affect the kinds of process behaviors it can support. These themes are a selection of behaviors we observed across multiple creative practices, but are certainly not the only possible design choices or relevant themes for all creative practices.

A creative version control system might support a **palette** of versions rather than a linear progression. To do so, it should consider providing **rapid, parallel access to versions** without fully resetting the state of the workspace, as discussed by the New Media Artist, Creative Coder, and Animal Behavior Researcher. In domains where outputs are separate from the state information, such as creative coding, the outputs should be **visually accessible and directly linked to the version state**, as explored by the Generative Artist and Tapestry Weaver.

**Confidence and freedom to explore** are essential across practices. However, VCS may place a **lower priority on rapid reversion** in order to gain these benefits. As seen with the Industrial Designer, Violin Maker, and Performer, easy reversion may not be a necessary capability: version histories provide these benefits even when additional labor is required to return to an earlier state.

**Lower-fidelity records** may enable variation, spontaneity, and adaptation, as valued by the Director, Performer, and Violin Maker. Similar benefits are found in low-fidelity sketches and prototypes, which allow creators to easily try variations [5], and leverage the ambiguity of imprecise representations to make space for interpretation and re-interpretation [17, 51]. Since software is plastic and can **dynamically change representations**, such tradeoffs need not be permanent: one stage of the creative process, such as early ideation or rapid improvisation, may require lower-fidelity presentations of version data and be willing to trade off easy reversion, while a later stage of refinement might display the full, high-fidelity records to enable easy reversion. Information visualization often leverages plasticity to adapt to the right level of representation [19]; version control systems may similarly benefit.

Making version data **accessible and visible across longer timescales and multiple projects** can support personal reflection and reworking of ideas, as seen with the Tapestry Weaver, Generative Artist, and Academic. Visualizations for reflection have been highly fruitful in VCS, across software development, writing, and education [3, 13, 18, 30, 37, 54, 56]; such tools and frameworks can provide a groundwork for longer-term approaches.

Version control systems that could be redesigned to benefit from these approaches include VCS for code, but also version histories of collaborative text documents, spreadsheets, and design files. These tools are often used by creative practitioners, but currently rely only on similar paradigms to software VCS (Figure 2). Modifications to additionally support the paradigms of creative practice will better support the processes of creative practitioners. When tools better support their creative processes, practitioners may also be able to better integrate the existing collaborative benefits of version control into their workflows.

### 5.2 Material and Medium

The uses of version information in creative process is intimately tied to the material properties of the version, and the medium of the creative practice in which the version is utilized. The creative medium influences the choice of material for versioning; likewise, the material of the version influences the role it plays in the creative process.

In some cases, we see strong similarities with creative practices that share a medium. For example, the Performer and Director, who both work in physical performance, both value change

and flexibility in their work, and choose lower-fidelity representations of version histories. As the Industrial Designer and Violin Maker both work in physical practices, they must recreate artifacts to return to earlier versions through labor-intensive processes. However, similarities between different mediums and differences within the same medium reveal aspects of creative process that are separated from any specific creative medium.

Software engineers and creative coders, though working in the same material, have radically different paradigms of creative process and the role of version histories. Though code and physical performance are different materials, VJ'ing, or live-coding visuals to accompany music, requires spontaneity and liveness in much the same way as a physical comedy show. The Physical Performer gains liveness by excluding the visuals of her performances from her version history; the Creative Coder uses rapid creation of parallel versions to allow him to pursue many different directions during a single performance, but only reuses a small selection of key modules between performances. Despite the different materials – bodily performance and code – the values are similar. The Physical Performer and the Violin Maker both use low-fidelity capture to make space for variation in their work, despite working in different mediums and on different timescales. There is much to learn by considering approaches to version control across mediums and materials, as well as within them, and this paper represents a step towards cross-pollinating across diverse creative domains.

We may also find value in considering creative process where version histories are mostly unused. Unique among our participants, the Ceramicist almost entirely rejects version histories in his work. The Ceramicist is an artist and ceramics studio technician and has been working in ceramics for 21 years. He collects and uses history information only minimally, and only as required for grants and show materials. In his day to day creative process, version histories are irrelevant: the knowledge of how to throw the base shapes in clay is embodied expertise, and the designs he creates are put together in the moment, linked by a single continuous theme. In a creative process like the Ceramicist's, external tools for version history are unnecessary.

## 6 LIMITATIONS AND FUTURE WORK

While our interviews spanned a broad range of creative practices, this is not a comprehensive review of all version control needs and strategies. We have identified several fruitful approaches, but there may be additional insight to be gained from other domains. Additionally, VCS is deeply entwined with collaborative and social contexts, where we may find productive parallels for other domains: might history management tools for other creative domains find a parallel for "starter code", or share inspiration through public forums of history repositories?

Process is personal as well as domain based, and future tools may find adoption between domains as much as within them dependent on individual needs. Such needs may also vary based on context and culture. An office environment may have requirements for what information is captured, or value efficiency and accuracy over personal process. The behaviors we observed are tied not just to domain and individual, but grounded in context and culture as well.

In future work, it will be important to explore how to practically integrate these themes into digital tools. We intend to build tools that instantiate these themes and deploy them with creative practitioners in workshop settings. Such studies will also explore cross-pollination between disciplines and contexts: how do practices benefit when tools support helpful behaviors from other practices?

We also hope this paper inspires other researchers to explore how to support creative process with version control across new domains.

## 7 CONCLUSION

In this paper we explored how creative practitioners in a wide variety of disciplines use version information to mediate and support their creative processes. Version control systems provide powerful tools for managing history and supporting collaboration. In our data, we see that creative practitioners use some of these features, and reject or appropriate others in service of their creative process: approaching versions as a palette of materials, gaining confidence to explore by capturing history, choosing varying levels of fidelity to capture version information, and reflecting and re-using versions over long time spans. Version control systems that are sensitive to these uses of version control information in creative process may provide large benefits to creative practitioners, and bring the collaborative benefits of VCS into creative workflows. We envision a future of widespread version control tools that are not just record keepers, but are collaborative partners intimately tied with the creative practice, bringing benefits to software engineering as well as a diverse range of creative domains. Such Creative Version Control Systems will be sensitive to the paradigms of specific creative practices and foreground the value of version histories to process.

## ACKNOWLEDGMENTS

We thank the anonymous reviewers for their support and feedback, and our interview participants for sharing their time and expertise. This research was supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. DGE 1752814.

## REFERENCES

[1] Teresa M Amabile. 2018. *Creativity in context: Update to the social psychology of creativity.* Routledge.
[2] Teresa M Amabile and Julianna Pillemer. 2012. Perspectives on the social psychology of creativity. *The Journal of Creative Behavior* 46, 1 (2012), 3–15.
[3] Thomas Ball, J.H. Kim, Adam Porter, and Harvey Siy. 1997. If your version control system could talk. In *ICSE Workshop on Process Modelling and Empirical Studies of Software Engineering.*
[4] Hugh Beyer and Karen Holtzblatt. 1999. Contextual Design. *Interactions* 6, 1 (1999), 32–42. https://doi.org/10.1145/291224.291229
[5] Bill Buxton. 2010. *Sketching User Experiences: Getting the Design Right and the Right Design.* Morgan Kaufmann, Amsterdam.
[6] Scott Chacon and Ben Straub. 2014. *Pro Git.* Apress.
[7] Kathy Charmaz. 2006. *Constructing grounded theory: A practical guide through qualitative analysis.* Sage.
[8] Kathy Charmaz and Linda Liska Belgrave. 2007. Grounded theory. *The Blackwell encyclopedia of sociology* (2007).
[9] Hong-Tai Chou and Won Kim. 1986. A Unifying Framework for Version Control in a CAD Environment. In *Proceedings of the 12th International Conference on Very Large Data Bases (VLDB '86).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 336–344.
[10] Marianela Ciolfi Felice, Sarah Fdili Alaoui, and Wendy E. Mackay. 2018. Knotation: Exploring and Documenting Choreographic Processes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18).* Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3174022
[11] Stephen M. Coakley, Jacob Mischka, and Cheng Thao. 2014. Version-Aware Word Documents. In *Proceedings of the 2nd International Workshop on (Document) Changes: Modeling, Detection, Storage and Visualization* (Fort Collins, CO, USA) *(DChanges '14).* Association for Computing Machinery, New York, NY, USA, Article 2, 4 pages. https://doi.org/10.1145/2723147.2723152
[12] Peter Dalsgaard. 2017. Instruments of inquiry: Understanding the nature and role of tools in design. *International Journal of Design* 11, 1 (2017).
[13] Dirk Draheim and Lukasz Pekacki. 2003. Process-centric analytical processing of version control data. In *Sixth International Workshop on Principles of Software Evolution, 2003. Proceedings.* 131–136. https://doi.org/10.1109/IWPSE.2003.1231220
[14] Alexandre Azevedo Filho, Ethan V. Munson, and Cheng Thao. 2017. Improving Version-Aware Word Documents. In *Proceedings of the 2017 ACM Symposium on Document Engineering* (Valletta, Malta) *(DocEng '17).* Association for Computing Machinery, New York, NY, USA, 129–132. https://doi.org/10.1145/3103010.3121027

[15] Jonas Frich, Michael Mose Biskjaer, Lindsay MacDonald Vermeulen, Christian Remy, and Peter Dalsgaard. 2019. Strategies in Creative Professionals' Use of Digital Tools Across Domains. In *Proceedings of the 2019 on Creativity and Cognition* (San Diego, CA, USA) *(C&C '19)*. Association for Computing Machinery, New York, NY, USA, 210–221. https://doi.org/10.1145/3325480.3325494

[16] David A. Fuller, Sergio T. Mujica, and José A. Pino. 1993. The Design of an Object-Oriented Collaborative Spreadsheet with Version Control and History Management. In *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice* (Indianapolis, Indiana, USA) *(SAC '93)*. Association for Computing Machinery, New York, NY, USA, 416–423. https://doi.org/10.1145/162754.162950

[17] William Gaver, Jacob Beaver, and Steve Benford. 2003. Ambiguity as a Resource for Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA) *(CHI '03)*. Association for Computing Machinery, New York, NY, USA, 233–240. https://doi.org/10.1145/642611.642653

[18] Louis Glassy. 2006. Using Version Control to Observe Student Software Development Processes. *J. Comput. Sci. Coll.* 21, 3 (Feb. 2006), 99–106.

[19] James D. Hollan and Benjamin B. Bederson. 1997. Information Visualization. In *Handbook of Human-Computer Interaction* (2nd ed.), M.G. Helander, T.K. Landauer, and P.V. Prabhu (Eds.). North Holland, Chapter 2.

[20] Ghita Jalal, Nolwenn Maudet, and Wendy E. Mackay. 2015. Color Portraits: From Color Picking to Interacting with Color. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 4207–4216. https://doi.org/10.1145/2702123.2702173

[21] James C Kaufman and Ronald A Beghetto. 2009. Beyond big and little: The four c model of creativity. *Review of general psychology* 13, 1 (2009), 1–12.

[22] Mary Beth Kery, Amber Horvath, and Brad Myers. 2017. Variolite: Supporting Exploratory Programming by Data Scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1265–1276. https://doi.org/10.1145/3025453.3025626

[23] P Yu Khudyakov, A Yu Kisel'nikov, I M Startcev, and A A Kovalev. 2018. Version control system of CAD documents and PLC projects. *Journal of Physics: Conference Series* 1015 (2018), 042020. https://doi.org/10.1088/1742-6596/1015/4/042020

[24] Joy Kim, Maneesh Agrawala, and Michael S. Bernstein. 2017. Mosaic: Designing Online Creative Communities for Sharing Works-in-Progress. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (Portland, Oregon, USA) *(CSCW '17)*. Association for Computing Machinery, New York, NY, USA, 246–258. https://doi.org/10.1145/2998181.2998195

[25] Scott R. Klemmer, Michael Thomsen, Ethan Phelps-Goodman, Robert Lee, and James A. Landay. 2002. Where Do Web Sites Come from?: Capturing and Interacting with Design History. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Minneapolis, Minnesota, USA) *(CHI '02)*. ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/503376.503378

[26] Ali Koç and Abdullah Uz Tansel. 2011. A Survey of Version Control Systems. In *The 2nd International Conference on Engineering and Meta-Engineering*. International Institute of Informatics and Systemics.

[27] Steinar Kvale. 1995. The social construction of validity. *Qualitative inquiry* 1, 1 (1995), 19–40.

[28] Bruno Latour. 1994. On technical mediation. *Common knowledge* 3, 2 (1994).

[29] Jingyi Li, Sonia Hashim, and Jennifer Jacobs. 2021. What We Can Learn From Visual Artists About Software Development. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 314, 14 pages. https://doi.org/10.1145/3411764.3445682

[30] Mark Mahoney. 2012. The Storyteller Version Control System: Tackling Version Control, Code Comments, and Team Learning. In *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity* (Tucson, Arizona, USA) *(SPLASH '12)*. Association for Computing Machinery, New York, NY, USA, 17–18. https://doi.org/10.1145/2384716.2384725

[31] Mark Mahoney. 2017. Collaborative Software Development Through Reflection and Storytelling. In *Companion of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (Portland, Oregon, USA) *(CSCW '17 Companion)*. Association for Computing Machinery, New York, NY, USA, 13–16. https://doi.org/10.1145/3022198.3023268

[32] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-Rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 72 (2019), 23 pages. https://doi.org/10.1145/3359174

[33] Brad A. Myers, Ashley Lai, Tam Minh Le, YoungSeok Yoon, Andrew Faulring, and Joel Brandt. 2015. Selective Undo Support for Painting Applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 4227–4236. https://doi.org/10.1145/2702123.2702543

[34] Brad A. Myers, Stephen Oney, YoungSeok Yoon, and Joel Brandt. 2013. Creativity support in authoring and backtracking. In *Proceedings of the Workshop on Evaluation Methods for Creativity Support Environments at CHI* (Paris, France).

[35] Molly Jane Nicholas, Sarah Sterman, and Eric Paulos. 2022. Creative and Motivational Strategies Used by Expert Creative Practitioners. In *Creativity and Cognition* (Venice, Italy) *(C&C '22)*. Association for Computing Machinery, New York, NY, USA, 323–335. https://doi.org/10.1145/3527927.3532870

[36] Taylor Palmer and Jordan Bowman. 2020. 2020 tools survey results. https://uxtools.co/survey-2020 Retrieved October 26, 2021, from https://uxtools.co/survey-2020.

[37] Jungkook Park, Yeong Hoon Park, Suin Kim, and Alice Oh. 2017. Eliph: Effective Visualization of Code History for Peer Assessment in Programming Education. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (Portland, Oregon, USA) *(CSCW '17)*. Association for Computing Machinery, New York, NY, USA, 458–467. https://doi.org/10.1145/2998181.2998285

[38] Ei Pa Pa Pe-Than, Laura Dabbish, and James D. Herbsleb. 2018. Collaborative Writing on GitHub: A Case Study of a Book Project. In *Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing* (Jersey City, NJ, USA) *(CSCW '18)*. Association for Computing Machinery, New York, NY, USA, 305–308. https://doi.org/10.1145/3272973.3274083

[39] Santiago Perez De Rosso and Daniel Jackson. 2013. What's Wrong with Git? A Conceptual Design Analysis. In *Proceedings of the 2013 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software* (Indianapolis, Indiana, USA) *(Onward! 2013)*. Association for Computing Machinery, New York, NY, USA, 37–52. https://doi.org/10.1145/2509578.2509584

[40] Marc J. Rochkind. 1975. The Source Code Control System. *IEEE Trans. Softw. Eng.* 1, 1 (March 1975), 364–370. https://doi.org/10.1109/TSE.1975.6312866

[41] Sebastian Rönnau, Jan Scheffczyk, and Uwe M. Borghoff. 2005. Towards XML Version Control of Office Documents. In *Proceedings of the 2005 ACM Symposium on Document Engineering* (Bristol, United Kingdom) *(DocEng '05)*. Association for Computing Machinery, New York, NY, USA, 10–19. https://doi.org/10.1145/1096601.1096606

[42] Nayan B. Ruparelia. 2010. The History of Version Control. *SIGSOFT Softw. Eng. Notes* 35, 1 (Jan. 2010), 5–9. https://doi.org/10.1145/1668862.1668876

[43] Hidenori Sato. 2019. Using grounded theory approach in management research. *Annals of Business Administrative Science* (2019).

[44] Donald Schön. 1983. *The Reflective Practitioner.* Perseus Books Group.

[45] Ben Shneiderman. 2007. Creativity Support Tools: Accelerating Discovery and Innovation. *Commun. ACM* 50, 12 (Dec. 2007), 20–32. https://doi.org/10.1145/1323688.1323689

[46] Anselm Strauss and Juliet Corbin. 1990. Open coding. *Basics of qualitative research: Grounded theory procedures and techniques* 2, 1990 (1990), 101–121.

[47] Lucy Suchman, Jeanette Blomberg, Julian E Orr, and Randall Trigg. 1999. Reconstructing technologies as social practice. *American behavioral scientist* 43, 3 (1999), 392–408.

[48] Lucy Suchman and Brigitte Jordan. 1990. Interactional troubles in face-to-face survey interviews. *J. Amer. Statist. Assoc.* 85, 409 (1990), 232–241. https://doi.org/10.1080/01621459.1990.10475331

[49] Michael Terry and Elizabeth D. Mynatt. 2002. Recognizing Creative Needs in User Interface Design. In *Proceedings of the 4th Conference on Creativity & Cognition* (Loughborough, UK) *(C&C '02)*. Association for Computing Machinery, New York, NY, USA, 38–44. https://doi.org/10.1145/581710.581718

[50] S. Turkle and S. Papert. 1992. Epistemological Pluralism and the Revaluation of the Concrete. *The Journal of Mathematical Behavior* 11 (1992), 3–33.

[51] Barbara Tversky and Masaki Suwa. 2009. Thinking with Sketches. In *Tools for innovation: The science behind the practical methods that drive new ideas.* Oxford University Press, 75–84. https://doi.org/10.1093/acprof:oso/9780195381634.003.0004

[52] The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics.* https://homotopytypetheory.org/book, Institute for Advanced Study.

[53] Chunhua Weng and John H. Gennari. 2004. Asynchronous Collaborative Writing through Annotations. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work* (Chicago, Illinois, USA) *(CSCW '04)*. Association for Computing Machinery, New York, NY, USA, 578–581. https://doi.org/10.1145/1031607.1031705

[54] Lisa Yan, Annie Hu, and Chris Piech. 2019. Pensieve: Feedback on Coding Process for Novices. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 253–259. https://doi.org/10.1145/3287324.3287483

[55] Nazatul Nurlisa Zolkifli, Amir Ngah, and Aziz Deraman. 2018. Version Control System: A Review. *Procedia Computer Science* 135 (2018), 408 – 415. https://doi.org/10.1016/j.procs.2018.08.191 The 3rd International Conference on Computer Science and Computational Intelligence (ICCSCI 2018) : Empowering Smart Technology in Digital Era for a Better Life.

[56] Fabio Zünd, Steven Poulakos, Mubbasir Kapadia, and Robert W. Sumner. 2017. Story Version Control and Graphical Visualization for Collaborative Story Authoring. In *Proceedings of the 14th European Conference on Visual Media Production (CVMP 2017)* (London, United Kingdom) *(CVMP 2017)*. Association for Computing Machinery, New York, NY, USA, Article 10, 10 pages. https://doi.org/10.1145/3150165.3150175

Received April 2021; revised November 2021; accepted March 2022