Live Writing the Live Coding Book

Alan F. Blackwell
University of Cambridge
afb21@cam.ac.uk

Geoff Cox
Aarhus University
gcox@cc.au.dk

Sang Won Lee
University of Michigan
snaglee@umich.edu

ABSTRACT

This paper is a speculation on the relationship between coding and writing, and the ways in which technical innovations and capabilities enable us to rethink each in terms of the other. As a case study, we draw on recent experiences of preparing a book on live coding, which integrates a wide range of personal, historical, technical and critical perspectives. This book project has been both experimental and reflective, in a manner that allows us to draw on critical understanding of both code and writing, and point to the potential for new practices in the future.

1. INTRODUCTION

There is something perverse in attempting to write a book about live coding[1]. Books are not particularly well suited to register the dynamic processes of how writing and coding operate as distinctive cultural-technical practices, unfolding in time. Coding would seem to be poorly expressed once printed on the pages of a book – although it is common enough for programming manuals and textbooks to do just this, encouraging the reader to read and type out the program in order to execute it. On the other hand the very constraints of printed form allow for other reflections to take place, while the textual qualities of writing code can be understood in the broader context of linguistic form - both on syntactic and semantic levels. Perhaps the disjunction in this project allows certain kinds of thinking to emerge that register how writing and coding are particular technologies that offer methods of inquiry and action[2].

Live coding gives us a way to think about coding again – what it can be, rather than what it is. For instance we can consider this use of the term as a technique of *defamiliarisation*, in the sense first intended by Viktor Shklovsky, as presenting familiar things in a strange way in order to enhance apperception[3]. Live coding makes coding strange again, allowing us to see beyond routine practices and interpretations of code. Of course, live coding also references a variety of alternative practices, including hacking, prototyping, code-bending and demoing, that are already recognised or understood to some degree. In this sense, it is not a wholly new phenomenon, but rather a critical orientation toward the otherwise conventionalised work of programming and software engineering (Bergstrom & Blackwell 2016).

In this paper, we wish to better understand this defamiliarising potential of live coding, by extending it into a parallel domain. We wish to explore the phenomenon of “live writing”, as a technique through which to think again about writing, and especially writing as enabled and mediated by new technologies. We have carried out this research in the context of a specific writing project: the preparation of a book about live coding, a project that we have been pursuing sporadically since the Live Coding Dagstuhl symposium (Blackwell et al 2014), and more earnestly at ICLC 2015 and at the 5th decennial Aarhus conference[4], as well as through various meetings of the Live Coding

---

[1] The writing project that provides the case study focus of this paper is being carried out in collaboration with Alex McLean, Thor Magnusson and Emma Cocker, together with others in the Live Coding community.

[2] For more on writing as a method of inquiry, see Laurel Richardson and Elizabeth Adams St. Pierre’s *Writing: A Method of Inquiry* (2005). We are grateful to one of the ICLC reviewers for this further reference.

[3] The term “defamiliarization” was introduced by *Shklovsky* in his (1917) essay “Art as Device” (sometimes translated as “Art as Technique”). For more on this concept and Russian Formalism, see, for instance, Fredric Jameson’s *The Prison House of Language* (1974).

[4] http://www.livecodenetwork.org/live-coding-alternatives/

Research Network (2014-16)[5]. Live writing could potentially be explored in the context of any writing project, not necessarily a book about live coding, although this particular project is convenient in that we have the analytic apparatus to hand that allows us to recognise the distinctive character of live writing by analogy to live coding.

Furthermore, this approach gives us an opportunity to test the generality that we hope might be achieved from the preliminary findings of our book-in-progress, by applying them outside of the coding domain to the parallel domain of writing. In presenting this work at the ICLC conference, we hope both to test the potential for broader insights from live coding research, and also to suggest a critical perspective on code, by thinking of the ways in which it is like writing[6].

2. WHAT CAN LIVE CODING TELL US ABOUT WRITING?

The central argument of this paper relies on drawing an analogy between the human practice of (software) coding on one hand, and the human practice of (academic) writing on the other. In presenting our argument to this particular conference audience, we are aware that most of those who read it are themselves familiar with both the practices of software development and of academic writing[7].

Unfortunately the very familiarity of coding and writing, both for ourselves and for our ICLC audience, presents challenges in the construction of an analytic argument. In our everyday practices as working academics, all of us tend to establish and maintain distinct working styles, ensuring that when we code, we code, and when we write, we write. In fact, as teaching academics, we often have to dissuade indifferent computer science students from the lazy assumption that writing a dissertation can be achieved through the same approach as writing a Java program. It is generally agreed that maintaining an appropriate separation between one's various thinking tools is a form of intellectual hygiene necessary for the development of academic skills.

In this project, we puncture that hygienic boundary. As already noted, live coding can be used as an analytic category through which to defamiliarise existing understandings of code. Here, we wish to defamiliarise the even more familiar understanding of how we write and think about writing. Later in the paper we will present specific experiments and analyses that undertake that defamiliarisation and begin to open up the literary and messy aspects of coding on one hand and the procedural and formal qualities of writing on the other. However, we are also able to draw on some precedents, in particular the emergence of the wiki as a writing tool that has also emerged from analogy to coding practices. Kent Beck and Ward Cunningham, original developers of the wiki, were primarily interested in the changing software development practices afforded by the new coding techniques of object orientation. It was clear that their philosophy of "agile" development was incompatible with the ways of writing presumed in the established "waterfall" contractual models of software development, in which documents acted as legal constraints on the functionality of the system.

The relatively agile writing style enabled by the wiki was therefore invented through direct and necessary analogy to the more agile coding style that had already been enabled by object oriented coding. Indeed, many technical aspects of the first wikis reflected patterns of OO languages – such as heavy reliance on reference links, the status of pages as independent objects in a heap unconstrained by narrative order, and so on. It is unsurprising that these innovations were established by Beck and Cunningham within a context of reflective practice, informed by thinking such as Christopher Alexander's Pattern Languages (Beck and Cunningham 1987).

Writing, like code, is both structured and emerges from structure. A common exhortation of computer science professors in the 1970s was that software should be written through a process of "iterative refinement". According to this model, it is first necessary to state, precisely, what the overall system is to achieve. Having done so, the author should then divide this overall goal into a

---

[5] http://www.livecodenetwork.org/about/

[6] We recognize that the parallel between writing and coding is not straightforward, and one cultural practice is not simply reducible to the other, but hope the reader is able to take the imaginative leap with us - in the spirit of live coding practice - and follow our speculative lines of inquiry.

[7] We also believe that our observations extend more widely, to other forms of coding and writing, although for the purposes of our current investigation will stay within these familiar realms.

set of components or sequence of operations. For each component, one states what it should achieve, in order to make the expected contribution to the overall goal. The same logic is then applied iteratively and recursively, until the operations are individual expressions of software code. These can simply be arranged in order, and the goal will have been achieved.

The top-down logic of software development by iterative refinement is quite different from (and contested against) the alternative logic of bottom-up construction. In this alternative model, individual parts are created and verified, then assembled to create a functioning whole. The spirit of the "bottom-up" developer is associated with that of the pragmatist or bricoleur, identified with craft rather than the rigorous proof considered to be desirable in the steps of iterative refinement.

The performance of live writing is always, to some extent, bottom-up. Words emerge from our mouths and fingers, as the units from which writerly concepts are abstracted and emerge. Plots, characters, arguments and analyses can be copied and extracted from a text, but for the writer, the diagram is not the thing. For a writer, a sketch is also a piece of writing. In this sense, writing appears always to be "live", in a sense that anticipates the innovations of live coding. It is only the printed residue of old writing that distracts us from that reality.

Live writing is overtly generative in this way, as it sets something in motion to produce unexpected results from simple rules. Nascent connections between generative computation and writing can be found in Formalist experimentation and the ways that systems might be understood more widely as devices to elicit wider transformations of form. The focus on generative processes and aesthetics are made explicit in Max Bense's theory of 'generative aesthetics' that drew together systems and information theory, semiotics and aesthetics to stress open-endedness of results (1971). He was partly drawing on Noam Chomsky's (1957) notion of generative or transformational grammar, in which grammar is assumed to be 'hard-wired' in human subjects. The focus on transformative grammar lends itself to the study of the formal qualities of code. This is not to say that similarities between formal and semantic features should be ignored, but that correlations be understood in an overall theory of all languages - both natural languages, and computational or other formalised systems. Although the grammars are complex, a better understanding of the separation of syntax and semantics and the transformations between them can help the writer, or live coder, predict possible outcomes and yet still be surprised by the unexpected outcomes that are generated.

Literary criticism, especially Russian Formalism and Structuralism, has examined and foregrounded these formal and procedural qualities of text, as if all writing simply followed prescribed structures, conventions, rules and grammars. This is consistent with the idea that language is not static or passive material but a kind of machine - as it endlessly transforms itself – and its parts cannot exist independently but only in relation to things in larger systems in which meanings are generated: "Every work, every novel, tells through its fabric of events the story of its own creation, its own history." (Tzvetan Todorov, from *Littérature et signification*, in Jameson 1972, p. 200). Writing does something in itself to the point where the written form seems to demonstrate a sense of autonomy from the act of writing – writing writes not writers, as Jacques Derrida would put it. In addition this seems to preempt advancements in machine learning such as the Dasher system that creates texts by predicting them (Ward et al 2002).

Needless to say there is a critical tradition here that would emphasise writing as material production and as centrally concerned with the political conditions of its making (e.g. see Fredric Jameson's *The Prison-House of Language* for a Marxist interpretation). Somewhat related to our method of defamiliarisation as presented in opening this paper is *Verfremdungseffekt* (alienation, or literally estrangement effect), a more overtly political tactic associated with Bertolt Brecht to examine the process of production of the text, one that reveals contradictions inherent in its form. The idea is to show the way things really work - to "lay bare the device" (in the words of the Russian Formalists), producing a disunity that is disturbing to the audience, and ultimately recognition that lived reality is similarly artificial and alienating. More precisely, knowledge of production leads in turn to knowledge of historical conditions. Live coding similarly draws attention to the technical apparatus or writing machine (laptops, source code editors, etc.) that produces it. The materiality of text or code is connected to the body of the writing machine – be it book or computer. It is also intimately connected to the body of the author and reader, and it becomes clear that writers, readers, texts,

coders, codes and machines all have bodies[8] and these are all in a sense live-coded inasmuch as they are produced in time and in public.

The relationship between speech and writing is key to these concerns, and both are further transformed by their relation to computation and coding practices. Contesting Derrida's notion that writing exceeds speech, N. Katherine Hayles has argued that code exceeds both speech and writing in its address to both humans and machines (Hayles 2005, p. 40). That live coders oscillate between these speech-like and written modes and practices leads us to speculate on whether live writing retains some of the unruly qualities of speech – coding as speech-act. This is important inasmuch as it establishes that speech is not free at all, but an act of legitimation of the ability to act freely and therefore ultimately an instrument of liberal governance. Moreover despite the established parallels of coding to speaking (Cox & McLean 2013), the use of the analogy to free speech in free software might be said to fall into the same trap - of libertarianism (Barbrook & Cameron 1996)[9]. Perhaps live writing is another way to highlight the unfree conditions under which communication operates and a way to begin to examine its effects. To write differently, developing new performative modes of expression becomes all the more important politically if we understand human subjects to be constituted in language[10]. And yet subjects – readers and writers – are not passive but also have the ability to reconstitute themselves through language, and even reconstitute the institution of language itself as if developed as part of a live coding performance.

To focus on the performative dimension of live writing as analogous to live coding calls into question some of the static assumptions of representational forms, and instead foregrounds the practices of representation, their effects and conditions. In the field of 'performance writing', for instance, there is a shift of attention from the practice of writing to the performance of writing – in speech and other technological forms – where existing discourses seem inadequate to capture the imaginative possibilities of what people and machines might do with language. Language constantly reinvents itself through social practice, cross-referencing its historical roots and possible futures through the inter-relations of speech and written forms. Writing has no beginning or end in this way as it opens itself up to its relations with existing works and its future extensions.

An important principle here is that language is material, and that all writing is the manipulation of this material into new configurations. In *Uncreative Writing* by Kenneth Goldsmith, for instance, this becomes clear in his call to not produce ever more new texts but instead to manage, organise and distribute them in divergent ways (Goldsmith 2011). He asserts that writers today begin to resemble programmers (as well as the other way around of course we would add), working with writing machines, generating and executing texts. In a co-authored text such as this we try something similar in assembling fragments, ideas, weaving together our voices with the voices of others in ways where originality no longer holds interest or relevance. This may appear disjointed at times but it should be expected inasmuch as the process imposes itself on the resultant text.

By analogy, the writerly and aesthetic qualities of computer programs begin to emphasise the materiality of language in similar ways. We are keen to emphasise the instability of intention and meaning here, and this is part of the attraction of live writing perhaps, to break out of the illusion of stability, to defamiliarise its codification. It is worth remembering that all writing is technology, and live writing like live coding manages to activate these slippages of meaning and material forms, and how writing subjects and objects become entangled.

3. EXPLORING LIVE WRITING AS SELF-EXPERIMENTATION

In a workshop devoted to preparation of the live coding book, the authors of this paper decided to experiment on themselves, through a process that would allow us to reflect on ourselves as "live writers", unsettling academic relations of first and third person, object and subject (McCarty 2015).

---

[8] By this we refer to the way in which bodies produce these entities as part of social practices of writing/coding, and how there is a body in the text/code and the code/text is a body.

[9] We are referring to the Free Software Foundation's clarification of their use of the term 'freedom' as aligned to free speech not free beer (http://www.gnu.org/philosophy/free-sw.html).

[10] This is one of the assumptions of Post-Structuralism for instance. See Louis Althusser's concept of 'interpellation' and the work of Judith Butler amongst others (Cox & McLean 2013),

Inspired by Sang Won Lee's presentation at ICLC 2015 (Lee and Essl 2015), we realised that the "performance" of the writing process could itself be captured and reviewed through the use of instrumented tools. We wanted to register the ways in which those tools might become part of the performance of writing, in a similar way to how coding is necessarily part of its own instrumentation. We therefore modified the structure of the workshop, to combine more conventional book-planning activities (drafting chapters, discussing themes, reviewing proposals) with explicitly live self-experiments.

3.1 Disruption with Wikis

In advance of the workshop, we prepared a modified version of the Gollum wiki editor.[11] The `<textarea>` on a web browser is modified to have extended functionalities that the Live Writing API[12] offers: recording and replaying writing process in real-time (Lee and Essl 2015). All kinds of a writer's interaction apparent inside the editor – not only individual keystrokes made to the editor but also other kinds of subtle activities, such as copy and paste, undo and redo, formatting, viewport-scroll, and highlighting – are recorded with timestamp information. Later, each session of writing a new wiki page (or editing an existing page) can be replayed in real-time. The replay reveals the dynamic process that would no longer be apparent in reading the final static text. First, the lives of words and sentences that have been written but later cut out are captured in the replay and shown to the readers. For the wiki editor, while each edit is automatically committed to a Git repo, the livewriting replay serves as the presentation of real-time morphing process between Git commits. Second, the Live Writing replay presents the pace and rhythm of a writer that may reflect the writer's states; contemplation, hesitation, or confidence. Lastly, the replay shows the nonlinear order of content generation. The reader watching the replay will be able to read the page in the order of the writer's thought process reflected in writing (as opposed to from top to bottom direction).

The first experiment was a "triadic experiment" in live writing, involving the three participants[13]. It was conducted in three rounds, each of which involved two participants working as a 'dyad', while the third became the 'commentator'. The dyad participants engaged in conversation, supported by writing. Meanwhile, the commentator listened and recorded his or her own interpretation of that conversation. Each member of the triad wrote their text in a node of Lee's Live-Writing extended Gollum wiki so that the temporal dynamics of the round could be replayed concurrently

One of the objectives of this experiment was to think through the relationship between speaking and writing, both as performative acts and temporal embeddings. It quickly became apparent that one does not speak and write simultaneously – their natural rhythms are incompatible, as is the register of voice required. Writing, even in a wiki, is no longer linear but a series of recursively embedded corrections and extensions. As a result, one member of the dyad would speak, while the other wrote, with these roles reversing with each conversational turn. Live writing thus became a mode of hearing, rather than a mode of speaking. The commentator synthesised, to some extent documenting the turns, but also framed the dialogue as dialogue, noting turns and devices. Individual voices were characteristically recognisable, one participant capturing words, another making statements, and the commentator alternating between documentary and critical responsibilities moving fluidly across subject-object positions. Witnessing the strong connection between speaking and writing in a collaborative session suggests the potential for extending the API to support sound recording in sync with the Live Writing replay. The audio augmented Live Writing playback (or Live Writing augmented voice recording) potentially reveals the process of a commentator's translation of dialogue into the text and might allow other participants to revisit and find ideas indexed across modalities.

3.2 Capturing through Recording

The second workshop experiment contrasted the collaborative writing of the first against individual voices exploring a common theme. After a day of discussing and planning the content of the

[11] https://github.com/gollum/gollum/wiki

[12] https://github.com/panavrin/livewriting

[13] The workshop participants were Emma Cocker, Geoff Cox and Alan Blackwell.

planned book, we wanted to assess the ways in which our thinking had developed, evaluate the extent to which we had developed a shared set of concerns, and explore the ways in which those concerns might be expressed in either distinctive individual or collective voice. We allocated 45 minutes to draft a new abstract for the book, moving beyond the publisher's proposal that had previously been prepared, but each writing independently. We created three new Gollum wiki pages, each of which would contain an individual abstract, and drafted those abstracts separately but concurrently, recording the sessions with the Live Writing API as before.

At the end of the day we set these three session recordings running, each on our own laptop, so that we could all watch and comment on the distinctive voices and styles that we adopted. In particular, we noted the ways in which an experiential approach to academic writing can involve placing a specific word "in the mouth" (on the screen) to try out how it feels, and explore the theoretical implications it might be expected to bring, even before the detail of that theory has become fixed.

Tools to record live writing performances, such as Lee's Live Writing API, allowed us to revisit and review the keystrokes of the writer (Lee and Essl 2015). When we used this to review our own writing, as created during workshop sessions, some characteristic patterns emerged. One of these was the ineffability of the pause – no keystrokes occur, and no trace remains. Was the writer deep in thought? In conversation? Or gone to get a cup of coffee? The framing of these pauses presents a vista of potential interpretations, quite unlike an algorave performance in which the beat continues while the coder thinks. For the efficiency of purpose, the current replay of the Live Writing API includes ways to navigate in time as any video media player does. Later, the visualization of keystroke activity will be added to help readers anticipate and navigate the edits. A second observation was that chunks of words that are pasted to the editor obscure the process of live writing. The writer wants to grab some existing text from the past, wants to quote (or borrow) someone else's writing or may want to hide behind the screen temporarily in fear of exposure of live writing proficiency. The lack of keystroke logs in the pasted text limits the liveness encoded in it.[14] A final issue, as with live coding, is the challenge for the observer in understanding how these live keystrokes are situated within a wider structural context. Our experimental wiki quickly grew to dozens of pages, which were rapidly and frequently reorganized (analogous to refactoring of object-oriented code). Although our words were recorded, this structural manipulation presents a far greater challenge for a more engaged conception of liveness.

3.3 Reflecting on Conversation

Any habitual academic collaborator will be familiar with the way that collaborative academic writing requires conversation. Shared understanding of the subject involves exchange of ideas and reference points, testing terminology, speculating about the range and restriction within which particular concepts or analyses can be applied. Our own collaboration in the development of the live coding book, and indeed of this paper, has required long periods of speaking and listening, not just writing. Some aspects of these conversations become dead ends, not finding application within the shared venture. Others are recognised or repeated, becoming central concerns of the eventual written work, and an emergent collective voice.

Our reflective and consciously experimental attitude throughout the current project recapitulates some of the strategies from the Dagstuhl Live Coding meeting. The Dagstuhl seminars encourage use of a wiki by the participants, and we ourselves used this to capture and review those conversations throughout the Live Coding symposium. As in formal committee meetings, the secretary often wields greater power than the committee chair – determining what the outcomes of the meeting were, and recording the responsibilities of those who have been delegated to act as agents of the collective voice. In our own subsequent work, we hope that we have to some extent escaped this trap. The one constant has been that we all write. The conversational collaboration is transmuted into a conversation of the page.

---

[14] The increasing tendency to write programs by copy and paste is in the vanguard of this kind of writing practice. http://www.theallium.com/engineering/computer-programming-to-be-officially-renamed-googling-stackoverflow/.

3.4 Problematising Historic voices

The live coding book, as currently conceived, has a dual purpose. The first of these is to document and communicate the broader implications of live coding, as addressed at the Dagstuhl symposium, to a wider audience, extending to educational, engineering and media implications. The second is to develop and extend the theoretical themes that emerged in workshop discussions through a piece of theoretical writing that although relatively conventional, will be relevant to the next section of this paper, where we explore how those theoretical themes might be applied back to the problem of live writing itself. Before doing that, we consider the first purpose, which suggests an alternative approach to the writing task.

In our current conception of the structure of the live coding book, the first half of the book incorporates a series of "expositions" where the voices of the live coding community will be heard directly in their writing, and also, in parallel, a more conventional historical overview of the origins and development of live coding. In the expositions section, we do not currently plan to subject our friends to quite the same extent of meta-authorial experimentation as we do ourselves. This is in part through sympathy, but also because we are concerned that authentic voices may well be rendered less authentic through such intervention.

The historical section, on the other hand, does seem to be ripe for experimentation. We build on previous work by Alan Blackwell (2006, 2015), applying the historical sociology techniques of Donald Mackenzie (2001). This method combines extended semi-structured interviews with close reading of contemporary historical sources and interpretive commentary, developing thematic concerns through grounded analysis. Such (re-)interpretation is necessarily contentious, even provocative, through the ways in which it challenges established narratives of scientific communities and the individuals within them. The interplay of voices between historical actors, contemporary researchers, and interpretive analysts is so inherently live and performative that it seems to offer rich opportunity for experimentation with live writing tools.

We have therefore created yet another experimental environment and protocol, this time situated between the conventions of oral history and interview-based media profile such as the work of Lynn Barber (1998). Once again, we wish to explore the potential of experimental writing technologies to capture and explore the liveness in this process. We have decided to use Piratepad, an open source descendant of the Etherpad collaborative editor.[15] Members of the live coding book team will be interviewing prominent figures in the early live coding community, with the full proceedings of the interview captured as a chat-like session in a Piratepad document. As with our live-capture modification to the Gollum wiki, Piratepad supports replay of all keystrokes and edit operations. We intend to preserve these recordings, and reference them in the book itself, so that the interview "performance", with all its pauses, provocations or rephrasings, remains available as history-unplugged.

4. THE SHARED LIVENESS OF ESTABLISHED PRACTICE

The procedural qualities of writing and coding lend themselves to updates and sharing of common resources, allowing for ongoing modification of texts in public. There are established practices for this in software production such as pair programming or code review as well as other cultural practices that promote social exchange and versioning such as the use of software repositories like GitHub. In academic writing there are different social conventions such as peer review or less positively where authorship is more carefully protected or simply condemned as plagiarism (ignoring the rich 'uncreative' practices of remix and appropriation mentioned earlier). Divergent intellectual property regimes play out these varying positions – copy-right or left – enforcing the idea of works as fixed objects or dynamic fluid exchanges accordingly (Stalder 2005). We think it is probably clear by now where our sympathies lie.

Most professional academics will have experience of projects in which they collaborate in writing a book. The variety of social forms in book writing offer a naturally occurring taxonomic order that we can use to explore the kinds of liveness that might occur in the writing process. At one extreme is

[15] http://www.piratepad.net/

the traditional monograph – the voice of a single author, presented as if a long-form lecture. Intermediaries are involved, including commissioning editors, proposal reviewers, copy editors and indexers. However the performance that we imagine is shaped by our shared understanding – the shared academic initiation of the written dissertation, and the perennial academic joke – "How are you doing? / Writing a book. / Neither am I!". Whatever life exists in the monograph, it is proverbially endured in private suffering, acknowledged largely in the many dedications to loved but neglected children and spouses.

The edited collection is a far more lively and diverse affair. A diffident editor may have a purely administrative role, collating the papers submitted and accepted for presentation at a conference. The papers may originally have been oral hybrids, performed in live reading to an audience, or they may be detailed scientific reports laboured over to gain reputation points in selective status competitions. A different kind of liveness is found in the many invited workshop proceedings, common in the social sciences, that establish a new community of interest or identify and explore a new theoretical topic. As with the Dagstuhl symposium proceedings that form part of the background to our own project, the vigour of the event can often be detected in remnants of liveness revealed in the written words.

Consciously thematic edited works involve increasingly lively interplay between the text of the editors and of the authors. Common conventions are the editorial top and tailing of alternative authorial voices, or section prefaces that assert the logic and intention of the volume. Extended editorial introductions are often relatively transparent, as a kind of operatic overture that both reveals the pleasures to come, and also expresses the optimism of the editor who hopes that a coherent whole has been achieved. More extreme collaborations include those coordinated volumes that construct a through-argument from many voices, often requiring multiple rounds of revision and cross-citation, to an extent that the exhausted editor (and authors) wish that they had embarked on a monograph instead.

Our own reflective experiments in writing process, informed by understanding of live coding, have attempted to introduce new kinds of liveness to collaborative writing, and recognise new kinds of media arrangement and the social relations that arise from this. Geoff Cox's collaboration with Alex McLean in *Speaking Code*, informs some of the ideas here but also the collaborative process in which writing and coding comingle as thoughts on the page. And of course, our own work on the live coding book as addressed in the rest of this paper are entangled in these ideas and how to best express them as emergent forms. Our writing has been undertaken through live experiments, as described in the previous sections. Those findings enable process observations from live coding to be applied or explored through writing. And this paper itself, as a reflective product of the book-construction experience, is itself a live product, growing out of the collaboration between the authors, and produced in the moment in the form of emails sent between live coding researchers who share some perspectives and inclinations with respect to code and media.

5. APPLYING LIVE CODING PERSPECTIVES TO WRITING

In drawing attention to the writing of a book still in progress and yet to be completed, we are in danger of disappearing into an endless loop of reflexivity. Yet this is not meant to be solipsistic, but rather a way of working through some of the technological constraints that allow for us to think and write at a particular moment in time, share our ideas in public and thus to contribute to the development of our ideas. The time aspect of this is particularly complex in that we write about the writing as we write, in parallel to the way that live coders edit code as it runs. In this way examining live coding and live writing together confuse strict definitions between them but also between past, present and future states, between the act of writing and its execution, allowing us to rethink some of our pre-existing notions of notation, liveness, temporality, systems and knowledge (that are the provisional focus for later chapters of the book). These concepts become ongoing conversations, inasmuch as all books are provisional and open to multiple interpretations and further modification.

The book will develop these ideas in more detail, and this paper will no doubt inform the development of one of its specific chapters. We want the book to be attentive to both its own literary and technological form in a similar way to *Writing Machines* by N. Katherine Hayles: "When a

literary work interrogates the inscription technology that produces it, it mobilizes reflexive loops between its imaginative world and the material apparatus embodying that creation as a physical presence." (Hayles 2002, p. 25) We aim to achieve something along these lines – establishing feedback loops between what is written and the conditions of its writing. In the book, and in this paper, we can interpret these livenesses by comparison to the live experience of music.

Live coded music has always invited comparison with music notation. In a naïve analysis, music notation might be considered as a specification of the music that is to be played, in the same manner as a program is a specification of the operations to be carried out by a computer. Alternatively, music notation might be regarded as a comparatively archaic recording medium, only approximately accurate in capturing the original (perhaps mental/internal) performance of an improvised creative composition. Music in the era of recorded sound has been bisected in the wake of Nelson Goodman (1968) into allographic "classical" score-based performance and autographic recorded improvisations from genres such as rock and jazz. Live coding challenges this bisection of practice, by being both score-based and improvised. The performance imperative to "show us your screens" as expressed in the TOPLAP manifesto makes it clear that neither sound nor performance are sufficient – live coding is about notation and laying bare the device of its making.

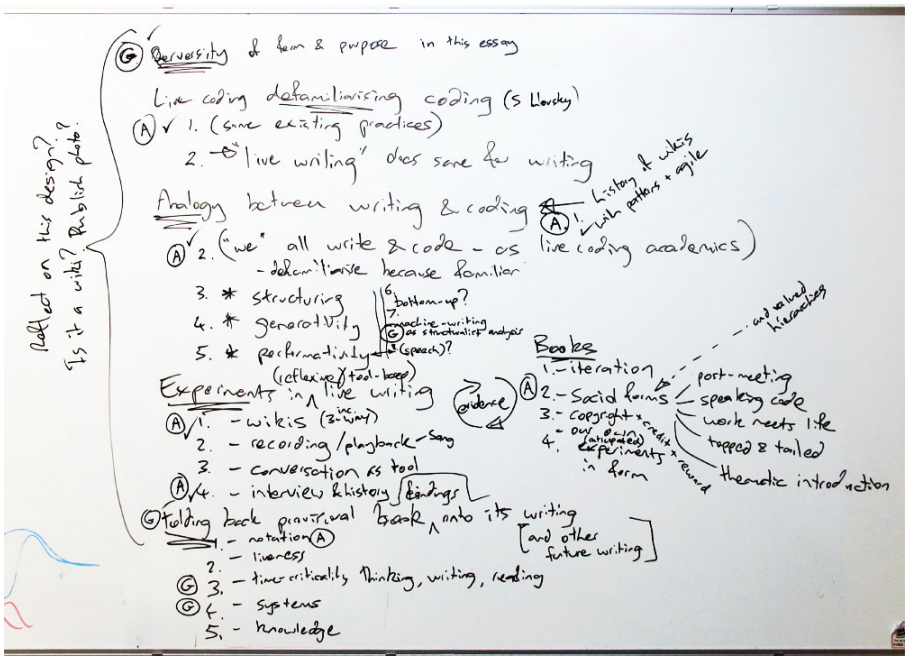

Figure 1. A snapshot of the live diagram guiding the structure of this paper.

In what ways might these particular characteristics of live coding throw light on live writing? Just as in the case of music notation, there is an obvious and naïve relation between writing and performance. According to this relation, the written work is a score, to be performed by a reader – either reading aloud in public or with an interior voice. However, just as it is naïve to regard music notation as a specification of sounds to be made, so it is naïve to regard writing as purely or necessarily a specification of words to be pronounced. Even in books, readers pause, return, or skip ahead. The notational conventions of a written page contain many elements beyond the words: margins, titles, numbers, fonts, lists, tables, equations or figures. As we improvise our book and this paper, we have "written" (drawn) both whiteboard diagrams and paper collages as shared artefacts of intention and effort (Figure 1 and 2). The tools of digital writing are themselves texts (file structure, editing menus, button labels), and we might decide whether to show them to the audience, just as generative composers sometimes project a real-time score that is visible to the audience and musicians at the same time. Writers read as they write, and the live coder/writer may also be reader, sharing the experience of the screen with readers in the audience.

There is far more to be said here, and too much for this paper. The "patterns of experience" identified by Blackwell (2015) offer many descriptions that are equally evocative of the writing performance. When we open up the process of writing to an audience, whether on a projected

screen, a whiteboard, a collaborative editor or simply a laptop shared on a table, serious consideration of the written word as a dynamic occupant of its new electronic home must surely follow.

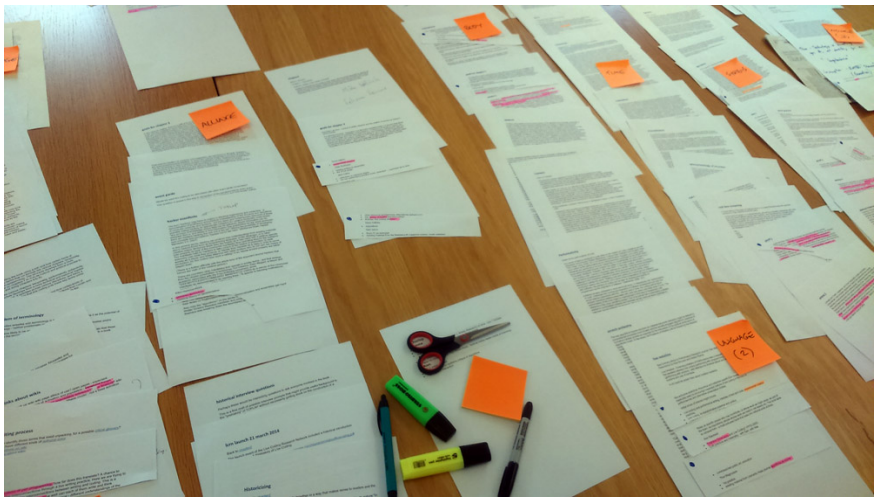

Figure 2. A collage of book fragments, linearising a wiki.

The method of this paper has been to think about the future of writing, by doing writing. We have reported on the writing of a book, and of course we have written this paper. In doing so, we have taken some care to avoid the obvious navel-gazing that might result from reflecting on a paper whose intention is to reflect on itself. Nevertheless, this fact must be acknowledged, both as an obvious implication of our thinking, and in order to make best use of the empirical resources that come most readily to hand. The live coding book is a work in progress. This paper, on the other hand, is nearly complete and live. What does it tell us about live writing as live coding, and where is the future of live coding as live writing directed?

If the liveness of this is reflected in the way that live coding is characterised by the ability to edit the code whilst running - coding during execution - then live writing is being performed now as we write (but not when you read these words sometime later). Liveness in software is usually associated with the ability to interact with a running system that is not stopped while waiting for new program statements. The ontological condition of liveness is extended by live coding in ways that media and performance studies has yet to conceptualise. If Philip Auslander, in his book *Liveness* (2008), outlined how mediatized performance has been traditionally granted authority through its reference to the live or real event, liveness has now even more fully incorporated into its mediatized forms and performance is less and less outside of its technological conditions. As mentioned previously, ‘performance writing’ makes an excellent example of this tendency where the performative character of writing itself is part of the work. Live events have come complex assemblages of humans, machines and codes, and we would say that live coding demonstrates some of the ontological confusions that characterise contemporary experience.

We are interested in establishing what this does to our understanding of live writing and how particular kinds of liveness are being produced at the moment of inscription. Live coding presents a direct challenge to the conventional understanding of the experience of time both with respect to the role of the programmer and user but also crucially through the way the machine understands time, whether in a wiki edit, a Git push, a keystroke log, a page render or a print queue. In live coding, these senses of time become entangled, placing conceptual configurations in the moment of their interpretation, subverting the mechanical determinism of the system clock. The temporality of computation challenges many of our presumptions of how we consider time to be structured and how we write and think in time.

In the case of coding, real-time operations and recursive functions such as loops offer alternative epistemological perspectives and social imaginaries. In his paper “... Else Loop Forever”, Wolfgang Ernst develops this discussion in relation to what he calls “untimeliness” (Ernst 2009). Contrary to the traditional musical performance – with beginning, middle and end – Ernst points out that the computational musical recording can be replayed endlessly: “with no internal sense of ending”. The

manipulation of musical time is key to this – the performance of the medium of time itself perhaps – where discrete events usually ordered into a sequence for a defined duration are instead open to nonlinearity and entropy. Live writing holds similar potential especially when discussing a larger project that is started but yet to be completed. Any sense of beginning or end is only a temporary holding position in lieu of further thoughts or publishing deadlines.

If the reflection on live coding through live writing is one way of relinquishing control (or exerting a different kind of control), then it is perhaps worth considering the broader context of systems that appear more and more opaque to us (and beyond our direct control). For instance, algorithms, that increasing seem to manipulate and control these systems, are not fixed but emergent and reactive phenomena, subject to constant 'live' updates. They are also part of larger socio-technical assemblages and infrastructures that are constantly evolving and subject to variable conditions and contingencies which include the active contributions of coding or writing. Algo-raves resonate with these issues and we will take up this issue in more detail in the book.

More generally what is at stake is a deeper way into the ways that space and time are constructed when coding and writing, and maintaining the knowledge practices of the academy in the face of corporate data infrastructure. All are open-ended performative processes, that attempt to reject deterministic or previously held certainties over the ways that meanings are generated. This perhaps provides a space for agency in human and nonhuman entities, produced through a fuller understanding of the operations of material-discursive systems we are written/coded into. In the case of the book we are writing, it partly writes us.

The experimental practices of live coding point toward a future conception of professional practice more appropriate to an engagement with machines as language-systems. This know-how becomes a way of understanding how code writes and is written, and how knowledge is being produced across human and machine registers. Our suggestion is that these live inter-actions of code and the practice of coding, defamiliarise knowledge production in this way, and expose how making and thinking might escape familiar forms and normative applications of all writing.

6. CONCLUSIONS

The ultimate objective of this investigation is not to make a retrospective analysis of writing practices as they have come to be, but rather to imagine the writing practices of the future. Despite our attempts at reflective self-experimentation, we are fully aware that the enterprise of writing and publishing the live coding book, and of convening and attending the Dagstuhl symposium from which it originated, are wholly embedded within the logic of the 20th century academic reputation economy. We are also aware that this logic is technologically contingent, and that future practices of the academy will be different, at least as much as the academy of the post-Gutenberg era differed from medieval manuscript-based scholarship.

We are confident that the underlying dynamics of our analysis – that writing is a practice of the notated voice, and that scholarship is an exercise of community – will continue to apply in the coming era. Nevertheless, the forms in which these dynamics are expressed seem likely to undergo radical change. New academic publishing conventions already borrow from the technical logic of software engineering (open access from open source, variational editions from source code configuration systems and so on). In the same way, since the intellectual frontiers of digital technology reflect the codes with which such technologies are constructed, so we expect that the innovative practices of live coding provide an essential lens through which we might anticipate the future of the book and the ways we read, write and think differently.

In this paper, our investigations have focused mainly on relatively mature software technologies – wikis, repositories and collaborative editors. Newer forms of image-based and network-intensive social media seem likely to emerge in new coding practices before they are embraced by academic scholars. How will rising generations of live coders have their voices modified by the enculturation of StackExchange, or Minecraft, or Twitter, or Slack or Snapchat? Serious consideration of such questions is likely to come first among innovation-seeking coders, and only later among academics for whom the surface traditions of scholarship represent security and status. We recognise that

there is a future for the book yet to be written. But we also believe that future will look a lot more like live coding than most people understand or can imagine.

Acknowledgments

We are grateful to Lucian Carata for configuration of the Gollum wiki used in our live writing experiments, to Emma Cocker for her contributions to the Live Writing workshop, and to Sang Won Lee's supervisor Georg Essl. Geoff Cox was able to spend research time at Darwin College Cambridge and the Cambridge Computer Laboratory with the support of an AUFF mobility grant and The Contemporary Condition supported by Danish Council for Independent Research.

REFERENCES

Auslander, P. 2008. *Liveness: Performance in a mediatized culture*. London: Routledge.

Barber, L. 1998. *Demon Barber*. London: Penguin.

Barbrook, R. and Cameron, A. 1996. "The Californian Ideology." *Science as Culture* 6(1), 44-72.

Beck, K. and Cunningham, W. 1987. *Using pattern languages for object-oriented programs*. Tektronix, Inc.Technical Report No. CR-87-43. Available online at http://c2.com/doc/oopsla87.html

Bense, M. 1971. "The Projects of Generative Aesthetics." In Reichart, J. ed., *Cybernetics, Art and Ideas*, London: Studio Vista.

Bergstrom, I. and Blackwell A.F. 2016. "The practices of programming." Manuscript under review.

Blackwell, A.F. 2006. "The reification of metaphor as a design tool." *ACM Transactions on Computer-Human Interaction (TOCHI)*, 13(4), 490-530.

Blackwell, A.F., McLean, A., Noble, J. and Rohrhuber, J. (2014). Collaboration and learning through live coding. *Dagstuhl Reports* 3(9), 130-168. Edited in cooperation with Jochen Arne Otto.

Blackwell, A.F. 2015. "Patterns of User Experience in Performance Programming." In *Proc. First International Conference on Live Coding*. Zenodo. http://doi.org/10.5281/zenodo.19315

Chomsky, N. 1957. *Syntactic Structures*, The Hague/Paris: Mouton.

Cox, G., McLean, A. 2013. *Speaking Code: Coding as Aesthetic and Political Expression*. Cambridge, Mass.: MIT Press.

Ernst, W. 2009. "'... Else Loop Forever'. The Untimeliness of Media." https://www.medienwissenschaft.hu-berlin.de/de/medienwissenschaft/medientheorien/downloads/publikationen/ernst-else-loop-forever.pdf

Goldsmith, K. 2011. *Uncreative Writing*, New York: Columbia University Press.

Goodman, N. 1968. *Languages of art: An approach to a theory of symbols*. Indianapolis: Hackett publishing.

Hayles, N. K. 2005. *My Mother was a Computer*. Chicago: University of Chicago Press.

Hayles, N. K. 2002. *Writing Machines*. Cambridge, Mass.: MIT Press.

Jameson, F. 1974. *The Prison-House of Language: A Critical Account of Structuralism and Russian Formalism* (Vol. 332). Princeton: Princeton University Press.

Richardson, L., St. Pierre, E.A. 2005. "Writing: A Method of Inquiry." Denzin, N.K., Lincoln, Y.S. eds. *The Sage Handbook of Qualitative Research* 3rd Edition. Thousand Oaks, CA: Sage, 959-978.

Lee, S.W. and Essl, G. 2015. *Live Writing: Asynchronous Playback of Live Coding and Writing*. Zenodo. 10.5281/zenodo.19322

McCarty, W. (2015). "A Matter of Style." *Interdisciplinary Science Reviews*, 40:2, 95-100

MacKenzie, D. 2001. *Mechanizing proof: computing, risk, and trust*, Cambridge, MA: MIT Press.

Stalder, F. 2005. *Open Cultures and the Nature of Networks*. Berlin: Revolver.

Ward, D.J., Blackwell, A.F. and MacKay, D.J.C. 2002. "Dasher: A gesture-driven data entry interface for mobile computing." *Human-Computer Interaction* 17, 199-228.