Evaluating programming systems design

Jonathan Edwards
jonathanmedwards
@gmail.com

Stephen Kell
University of Kent
S.R.Kell@kent.ac.uk

Tomas Petricek
University of Kent
T.Petricek@kent.ac.uk

Luke Church
Computer Laboratory
University of Cambridge
luke@church.name

Abstract

Research on programming systems design needs to consider a wide range of aspects in their full complexity. This includes user interaction, implementation, interoperability but also the sustainability of its ecosystem and wider societal impact. Established methods of evaluation, such as formal proofs or user studies, impose a reductionist view that makes it difficult to see programming systems in their full complexity and, consequently, force researchers to adopt simplistic perspectives.

This paper asks whether we can create more amenable methods of evaluation derived from existing informal practices such as multimedia essays, demos, and interactive tutorials. These popular forms incorporate recorded or scaffolded interaction, often embedded in a text that guides the reader. Can we augment such forms with structure and guidelines to obtain methods of evaluation suitable for peer review? We do not answer this question, but merely seek to identify some of the problems and instigate a community discussion. In that spirit we propose to hold a panel session at the conference.

1. Introduction

The ability to disseminate knowledge, which relied on the invention of the printing press, was essential for the birth of modern science in the 17th century[1]. The internet changed how we distribute academic papers, but their format has changed surprisingly little. Despite new media formats that the internet enables, our research is still largely presented in the same format as in the 17th century.

There are some noteworthy exceptions. An increasing number of academic publications that rely on computational methods publish "software artifacts" in addition to a paper which, at least in principle, allow others to repeat the experiments. Grossman et. al. (2016) propose embedding "animated figures" into research papers through multimedia features of PDF documents. Some authors create *interactive essays* or *explorable explanations* (Brusilovsky 1994, Victor 2011), which are articles that the readers can interact with in order to better understand the presented subject, for example by modifying input parameters of a computation and seeing how this affects the results. Those cover topics ranging from social science, e.g. (Hart and Case, 2014) to machine learning, e.g. (Carter et al., 2016), but they are typically educationally focused and do not present new research results.

We argue that we need to establish new media for presenting novel research ideas on programming systems design. We want to make such new media a core method for presenting some aspects of our work. Finally, to put such new media on a par with other ways of presenting ideas on programming, we want to find rigorous evaluation methods for such new media presentations.

Research on programmer experience needs to explore the capabilities that a system design offers to a programmer, how those affect programmers conception of the problem at hand and how it changes the way they conceptualize the world they need to model and interface with. Traditional static paper format is unsuitable for exploring such questions. A new media format we envision makes the reader

[1] As documented by Wootton (2015), the printing press made knowledge public and allowed a wide-spread exchange of ideas among, for example, astronomers belonging to different cultures.

more naturally ask the aforementioned questions and it is also more suited for discussing systems that go beyond the textual program representation.

The aim of this paper is not to present a final new media alternative to the static paper format, but rather, to start and stimulate a discussion on this topic. In order to do this, we present a range of different positions that are rooted in different backgrounds, have different motivations and argue for different methods. Although the focus of individual positions differs, we structure each statement in a similar way. We hope that this format adds a level of coherency between individual positions, but more importantly, will also encourage others to contribute to the debate that we are hoping to start with this paper and with a proposed panel session at the conference.

2. Multimedia essays: Evaluating experience design

One important kind of research is about *designing* programming systems to improve their human *experience*. We italicize the words *designing experience* because they are a problem: they make it difficult to report research contributions using traditional academic papers, for two reasons. First, papers restrict explanation of interactive experiences to static figures and screenshots. Secondly, the norms of academic papers are not well suited to discussing design issues, which fit neither into reporting empirical results nor proving theorems.

Is there a better way to report novel designs of system experiences? We propose to adapt an established informal practice: multimedia web essays (primarily text containing video clips) for this purpose. The challenge is to make this form rigorous enough to be used in a peer review process. Our approach to this challenge is to impose structure and guidelines, for both the authors and reviewers.

2.1 Why does it matter: Evaluating live programming research

Consider the nascent field of Live Programming (Tanimoto 1990, LIVE 2013). This field aspires to make programming more like "spreadsheeting" by synthesizing a visual substrate offering immediate concrete feedback at all times. Such research is a matter of co-design of a PL and IDE, rethinking the nature of both so that in combination they offer a live programming experience. This is not PL research, as it is not about formally verifiable properties. Neither is this HCI research, as it is not foremost about observable user behaviors, but rather about the conviviality of conceptual models and visualizations.[2] As a result it is difficult to publish research on Live Programming in either PL or HCI venues. More insidiously, publishing in such venues can lead us off course by forcing us to answer the wrong questions.

On the other hand, venues like the LIVE Programming Workshop (LIVE 2013) accept work presented informally, through screencasts and verbal presentations. But lacking written exposition that can be carefully reviewed, it can be hard to ascertain what contributions these submissions make. Also lacking discussions of related work, it can be hard to situate such contributions as a conversation among researchers, which we feel is crucial to making progress as a field.

We need to find some way to report our research that is suitable for discussion of system experience design while also being rigorous enough to support the proven norms of research: peer review, archival publication, and citation.

2.2 What are the problems: Evaluating design

Our goal is to improve the human experience of using and developing software. This goal is inherently: unformalizable and unquantifiable. Therefore the discussions we need to have are about *design*: discovering technical choices offering a propitious balance between conflicting and sometimes subjective qualities.

[2] We agree in principle that a mature Live Programming system should be validated by randomized controlled trials, but that is wholly impractical at this stage of the field, and we see little evidence that this method has succeeded even in more mature fields of software research.

We need a way to evaluate the design of systems from the viewpoint of their human experience, and we need this to be lightweight enough to discuss early-stage work. Video is the natural medium for demonstrating user experiences. But carefully crafted prose is still the best medium for explaining ideas in detail to be studied and reviewed by others. These requirements match an emerging form of discourse on the web: multimedia essays consisting of text with video clips. An influential early example of this form is Bret Victor's Learnable Programming (Victor 2012), but see also (Petricek 2016). These essays tend to be informal and introductory, addressed to an audience of interested observers rather than peer researchers within the field. To serve as a medium for reporting and evaluating research contributions, more rigor is required.

2.3 How to address those problems: Tentative guidelines

Every field of research adopts certain methods for reporting and evaluating contributions, and in many ways these methods define the boundaries of the field. These methods strike a contract (often unwritten) between authors and reviewers. We propose to adapt multimedia essays into a method of evaluation by making this contract explicit as a set of guidelines for both authors and reviewers, backed by normative examples.

Many forms of evaluation become rigorous by setting high standards for authors to substantiate their contributions. Instead we shift part of the burden onto the expert reviewers, who are asked to use their best judgement to answer the question: would it benefit the research community to publish this idea?

Guidelines for authors: What makes for a rigorous multimedia essay

1. The essay identifies a targeted domain of software, and a targeted profile of user/developer.
2. The essay describes the state of the art and enumerates specific contributions that are claimed to be novel solutions with benefits over known techniques. Related work is cited either in context or a separate section.
3. Each contribution is categorized as either a technical novelty in some field (eg PL) or as a novel design combining or unifying known techniques (within or across fields). Each contribution states what benefits it is claiming, categorized by taxonomies such as Cognitive Dimensions (Green&Petre 1996) or Olsen (2007, summarized by Hempel 2019).
4. Each contribution is discussed in terms of one or more authentic examples from the problem domain. The benefits over known techniques are substantiated by comparison with their application to corresponding examples. Benefits to user experience are demonstrated with video clips, where possible also for the comparable known technique.
5. Each contribution discusses its implementation: is the demonstrated experience a mockup or an implementation? What is novel about the implementation? What is the hard part? How general is it—how would it handle variations on the demonstrated scenarios?
6. Design deficiencies and unsolved problems are disclosed, preferably in terms of problematic examples, and especially problematic variations upon the examples used to claim benefits. As a rule of thumb, there should be about as many problematic examples as beneficial ones.
7. Discussion of mistakes made, lessons learned, and suggestions for future research are all encouraged.

Guidelines for reviewers: How to recognize important contributions

1. Unless the submission claims to be a survey, reviewers should only expect familiarity with significant related research, not encyclopedic knowledge.
2. New ideas can seem obvious in retrospect. Assuming that the essay shows general familiarity with the field, if a reviewer believes a contribution is trivial or already known then it is their duty to substantiate that position (see Olsen 2007).
3. Offer constructive criticism, about both form and content. In addition to technical issues, it can be very helpful to note unclear prose and the need for examples. Distinguish between two kinds of criticism: criticism of the evaluation methods used vs. criticism of the ideas themselves. Weight these two dimensions appropriately for the venue.

4. Answer this question: in your expert judgement, would it benefit the research community to publish this submission? In other words, is it worth reading?

2.4 What will this allow: Towards humane technology design

A better medium in which to report and evaluate our research will speed and tune the intellectual feedback loop that leads to progress, and sometimes even breakthroughs. In the longer term, establishing a distinct methodology can help us mature into a fully formed field. More broadly, we see ourselves as part of a wider movement toward *humane technology design*, and we hope that these methods might help others join together into productive research communities.

Next steps: Building examples and growing community

1. Solicit feedback from the community—indeed it is the purpose of this paper to initiate that conversation at the Psychology of Programming Interest Group (PPIG) meeting.
2. Do trial runs with our own work, leading to a more refined and detailed proposal.
3. Hold a workshop inviting submissions using our proposed methods, as well as meta-discussion of the methods.

3 Interactive essays: Asking new questions about programming

In this section, we present an argument in favor of *interactive essays*. This position shares many aspects with the previous one. However, it additionally requires interactivity that allows the reader to use some aspects of the presented system in some, possibly limited, ways. The interaction is guided by the essay author - the mixed content format allows the author to first introduce design principles and then lets the reader get a glimpse of what the envisioned experience looks like.

The most important aspects of this proposal are that, although the narrative is controlled by the author, the reader can experience some aspects of the system live and that the reader can also deviate from the narrative suggested by the author and explore the boundaries of the actual prototype or demonstration implemented by the author.

An example of such interactive essay is an introduction to the theory of coeffects by Petricek (2016). This example is more educational and it is centered around programming language theory research, rather than programming experience, but it follows the format that we argue for here: it allows the reader to interact with a (somewhat limited) system within a narrative designed by the author.

3.1 Why does it matter: New media for new thoughts

Present ways of publishing and evaluating programming research are not doing a good job:

- A paper presenting programming systems design research can easily be rejected for the lack of evaluation. This is not because the authors did not want to evaluate their work, but because we do not have established standards for evaluating such work.
- A paper focusing on interaction or user experience is often treated as "early work". This is because we assume that the idea is not yet mature enough to be, say, formalized and presented in the form of mathematical properties.
- Consequently, authors often change the focus of their work and start studying formal or empirical aspects that are easier to evaluate. In doing so, they abandon the aspects that made the work interesting in the first place.

A new media format that allows us to effectively present user experience or interaction with a programming system has the potential to change this. If we succeed at finding a way of rigorously evaluating such contributions, we will be able to focus on studying how programmers interact with systems and how new ways of interaction can better support programmers in their job. We will also be able to better understand and extend other work in this space, building a flourishing community of programmer experience researchers.

3.2 What are the problems: Rigorous communication and evaluation tool

To make interactive essays an effective research communication and evaluation tool, we need to change the perception surrounding interactive essays, find rigorous ways of reducing complexity of systems and tackle development and archival issues.

From education to research

Recent work on interactive essays has been focused on education teaching, for example, how a certain model or algorithm work. This is useful, but it does not address our needs. If we are to use new media to publish novel research ideas, we need to find a suitable method of structuring the ideas. An example of such structuring from formal methods is the idea of *structured operational semantics* (Plotkin and Kahn, 1987), which allowed a growing number of people to present new programming language constructs. Structured operational semantics defines a common way of reasoning about the correctness of a programming language, but it can be applied to topics ranging from concurrency to network communication. Similarly, we need a structure of interactive essays that defines a common method, but can be used for a wide range of programming experiences.

Finding new way of reducing complexity

When evaluating programming systems, we might hope to implement the whole system, use it for a non-trivial task in a real-world setting and reflect on the experience and wider societal implications of the system. This is infeasible and so we need to reduce the complexity of the task in some way.

Different evaluation methods ignore different aspects of such complexity. For example, formal proofs only look at small models ignoring all practical aspects. How can we simplify the presentation of a user experience so that we still capture important aspects of the desired user experience and, ideally, convey this to the reader or viewer in a direct way? Do we need to make our presentation interactive enough that a reader can feel what using the system would actually feel like?

Rigorous evaluation of user experience

In order to be able to evaluate such presentation of programming ideas, we also need to understand what counts as a rigorous presentation of a programming system in an interactive new media format. We have a good understanding for theorems involving mathematical properties and for empirical evaluation, but what are the standards of rigor for interactive illustrations of user experiences?

Development and archival issues

The final challenge of presenting user experiences in an interactive format is that, compared to a traditional paper, such presentations are significantly harder to develop and archive. There is a range of options. A static article with embedded screencasts (as discussed in Section 2) is relatively easy to produce, but it might be too guided to effectively convey the programmer experience. A partly functioning system that runs in a web browser is more complex and may convey the idea more clearly, but it is only usable for certain kinds of programming experience research. Finally, web-based essays will inevitably be more difficult to archive. This is easier to address if they are self-contained, but can get increasingly difficult if such essay uses external services or data sources.

3.3 How to address those problems: Methodologies, community and technologies

Making interactive essays an established way of presenting and evaluating programming system design poses technical, academic and social challenges. For this reason, addressing the problems outlined above requires new technologies, new academic standards and also a new community.

Extensive methodology section

It will take time until the community producing interactive essays settles upon standards of rigor. This will most likely take the form of tacit unwritten knowledge, as identified by Polanyi (2012). Until such standards are widely understood, we need to include an explicit and detailed methodology section in our publications. This needs to clarify what method the essay follows and which aspects of it should be evaluated in what ways. For example, when a new system design is presented, one can claim that it is interesting for its novelty. This can be supported by a comprehensive comparison with

related work. Alternatively, one might claim that a certain theory allows an interesting range of programmer experiences. This can be supported by a prototype implementation of some of those and an outline of work that needs to be done for other cases. By having an explicit methodology section, the readers and reviewers will know how the key contributions of the essay are supported.

What makes new media rigorous

An interactive essay should share many common properties with standard academic papers. It should have a methodology section that is often missing in computer science papers, discussion of related work and clear statement of contributions. An interesting problem is how to judge the rigour of the interactive aspects of the essay. We offer several suggestions:

- The essay should outline how can the proposed system design be implemented. Essays based on a prototype implementation should document what has been omitted and how it could be completed. Essays not including implementation need to argue in another way.
- The essay should illustrate some interesting "effect" such as a new, appealing user experience story. This can be documented by an interactive walk-through. A rigorous essay should also document how this experience arises from a more basic design principles and how those principles extend to other aspects of the system.
- In order to develop a flourishing community, system design essays need to have common points on which they can be contrasted. Those can be common programming challenges or domains - a rigorous essay will aim to include user experiences that make it possible to relate the presented ideas to existing work.

3.4 What will this allow: Changing the questions we ask

The way we evaluate our research has an effect on what research we conduct. According to present day standards, programming systems design research often appears as early work. It rarely comes with proofs, detailed empirical evaluations or large scale user studies. However, adding any of these often shifts the focus of the work away from the original focus on *programming systems design*.

If we can (i) make interactive essays a standard format for presenting programming system design research, (ii) build a community around such format and (iii) agree on standard of rigour for such essays, then we will be able to move the field of programming systems design forward. The format will make it possible to evaluate the design of such systems from a more comprehensive perspective. It will allow producing research publications that share enough of their problems and methods so that they can be compared and inspire others to contribute new ideas to the common ecosystem.

4 Papers plus: Letting the format suit the work

If we want to transcend the limitations of paper, we should be careful not to transcend its benefits too. In this section, we outline some of the benefits of traditional paper formats and discuss how new media formats can keep the good properties of well-written static papers.

4.1 Why it matters: Evaluable and durable formats for evaluation by inquiry

A clear motivation for exploring such alternatives is in the spirit of 'the medium is the message' (McLuhan, 1967) and the potentially helpful disruption that new media may bring. For example, inclusion of new media may usefully break ice between researchers and (other) practitioners. Lacking a formalised or "scholarly" culture, communicative practitioners exploring new ideas have already pursued a broad range of techniques, from web-based written documents that in some cases resemble papers, through video essays, interactive demos all the way to simply releasing software.

The primary advantages of most 'new media' approaches has been some mixture of explanatory effectiveness and support for 'evaluation by inquiry'—active investigation on the part of the reader or consumer, rather than simply a report by the authors. There is clear value to these, but also the clear challenge: how to make them sufficiently evaluable and durable that they might take a place alongside traditional papers.

4.2 What are the problems: holistic and versatile reporting

While gaining in explanatory effectiveness, we would also like to retain some abilities: to talk holistically and comparatively, and to deal in relatively evaluable artifacts. There is also a latent question of how prescriptive any submission format should be: work is diverse, so progress relies on enabling a sufficiently diverse range of 'shapes of submission'. Certain new risks also emerge: for example, papers can be anonymised and otherwise curbed in respect of the prejudices they might elicit, whereas, for example, audio and video present inherent problems if they are fronted by a human face or voice.

Versatility of academic papers

Papers are versatile: they offer a viable way to walk through arguments, methods, designs, experiences and other findings. They are inherently fairly accessible to readers (requiring little fancy software or equipment to experience), fairly durable, and extend to a huge range of topics, not only in subject matter but in modes of contribution (position papers, empirical studies, design rationales, surveys, and so on).

Conveying design holistically

One specific challenge in not-just-paper modes, especially anything resembling a demo, is to convey a design holistically. Programming systems research often has relatively abstract goals, such as to improve the comprehensibility or factoring of a certain kind of code, to improve the economics of a long tail of currently tedious programming tasks (e.g. integration), or to introduce some elaboration of an area that is currently poorly understood or ill-characterised. In respect of these, demo-like vehicles may struggle: since a demo focuses at all times on presenting something specific, it risks an over-narrow presentation which conveys one de-emphasises breadth. *Showing* a long list of examples, while possible, is harder than merely naming or describing them. Showing a smaller number and simply naming others (e.g., in narration) is perhaps viable in some cases. In others, a paper may still be the best option. Holism is difficult to account for: human beings' ability to generalise and to imagine is sometimes most effectively triggered by a well-conveyed example, but equally, details can distract from the big picture.

Retaining comparison

Since a demo has (one assumes) a unilateral focus on `the' artifact being discussed, it makes more difficult the comparative mode of exposition, on which most research papers rely heavily to convey both novelty and essence. The de-emphasis of comparators is already a risk in papers: a common pattern in poorly written papers about practical work is to focus on 'the thing we built' without addressing the question of how it differs from existing systems and hence why it should be accepted as novel and valuable research.

Realising one's comparators in a demoable form, while possible, is harder than naming or describing them. To make a comparative point, focus on 'a demo' of 'the artifact' must either be paused, or must embody the alternative—a juxtaposed 'demo within a demo'!—to handle comparative questions. Any expectation to do this may raise the effort bar unreasonably high. On the other hand, making this effort has obvious value in encouraging like-for-like comparison and reducing the risk of mischaracterising the prior work. In effect it amounts to a reproduction study.

Effort, benefits and costs of new media presentations

Another concern is balancing effort, benefit and credit. While achieving the above-mentioned 'difficult but possible' feats in demo-like form is certainly desirable in principle, the effort and reward may or may not be proportionate. Depending on the other contributions on offer, they may or may not fit into a reasonably sized and shaped 'unit of contribution'. One of the strengths of papers is that they allow some flexibility in these sizes, shapes and proportions. The goal should be to admit more flexibility, not less, while retaining a cost-effective process from submission to review to publishing. Just as 'possible, but difficult' suggest greater demands on the author, 'evaluation by inquiry'

arguably makes more demands of the reviewer, relative simply to reading a paper and digesting its argument (already a considerable task!).

4.3 How to address those problems: Presenting a structured argument

A possible guiding principle might be to oblige authors to present a structured argument as the primary object of evaluation, as with papers. Inquiry-based evaluation may be offered as an additional mode by which the evaluator may satisfy themselves of that argument. Authors who prefer to 'let the system do the talking', may do so to an extent, without pushing great responsibility for initiative onto the evaluator. A common ingredient in all forms of submission should perhaps be a linearised presentation of this argument---in short, it should be possible to 'press play', even if the artifact permits inquiry or interrogation. The more interrogatable an artifact, the more contribution is potentially apparent from that, as opposed to from writing.

In light of the issues of prejudice and anonymisation identified earlier, a policy of 'no voices, no visages' may be preferable (e.g., subtitles rather than voiceover).

In some communities, artifacts or tools by themselves are an object of independent recognition. Some conventions are therefore required on what is covered by a given submission versus what aspects may be redeemable later for 'extra credit' by a submission with different emphasis. The right conventions are unlikely to be easily anticipated; observing the emergent patterns is necessary.

4.3 What will this allow: Tailoring formats to the needs of the work

If done right, an approach will allow authors to tailor a format to the needs of the work, mixing some amount of new media (video, audio, web-like interactive features, or something yet unanticipated) while retaining a clear argument. If such a format can succeed, it may break the ice, without breaking the glass: allowing a wide range of work to gain credit and recognition in a publication-based system, from diverse participants and backgrounds, while still allowing fair and thorough evaluation. Accommodating a continuum, with papers remaining a point thereon, may prevent prevent dismissive attitudes or 'ghettoisation' of work choosing these new approaches.

5 Conclusions

When writing a programming system paper, we have various ways of evaluating our work. We can study its mathematical properties and publish proofs, we can measure our implementation and publish charts or we can run user studies and publish quantitative or qualitative summaries. A static academic paper is a great format for presenting such evaluation, possibly with a software artifact for empirical measurements. But are our preferred evaluation methods determining our publication format and research focus, or is our publication format and research focus determining our evaluation methods?

In this paper, we argued that we need to establish new media for presenting novel research ideas on programming systems design. Our aim has been to stimulate a discussion. To this end, we presented three different positions that are inspired by different research problems and argue for somewhat different methods. We discussed a range of challenges that such new media presentations are facing, possible ways of addressing those and our thoughts on how the world of programming systems research would benefit from adopting such new formats.

5.1 A range of technological solutions

We presented three concrete positions in this paper, arguing for media ranging from traditional papers, suitably augmented with other media to interactive web-based essay that run a prototype of the envisioned system directly in a web browser. One tentative conclusion of this paper is that we should accept and welcome a range of presentations including:

- An *interactive essay* combines text that provides a narrative in the similar sense in which a traditional academic paper does with interactive components that show aspects of the discussed system in action. This is an ambitious goal as it involves not only producing a prototype implementation, but also orchestrating the interaction with the reader in the essay.

- In a *screencast essay* with embedded videos, each screencast documents one particular interaction. It does not let the reader choose a different path, but it does let the reader see what the actual interaction with the programming system looks like. We should accept screencasts that are based on an actual implementation or mock-ups, but we need to acknowledge the difference as part of the methodology section included in each publication.
- An essay based on videos can be made more interactive by offering the reader a choice of several predefined alternatives as in *choose your own adventure* games. This is similar to wireframing tools known from user interface design, but we should not follow wireframing tools in the level of abstraction they use. Our goal is to provide full record of user experience.
- *Scrollytelling* (Seyser and Zeiller, 2018) refers to online articles that present a story, which unfolds as the reader scrolls through the article. Scrollytelling articles often combine text, where the reader just scrolls down, and interactive sections where scrolling takes a different role and allows the reader to move back and forth through an video-like visual element[3]. In its basic form, scrollytelling allows the viewer to easily go backwards and replay parts of "story" that they are interested in. It could also be combined with an interactive essay, which would let readers scroll to follow the author's narrative, while allowing them to take a different course of action to explore other aspects of the system.
- *Onboarding* refers to a range of UI design patterns for training new users with minimum friction (UI-patterns 2019). These patterns scaffold or restrict the UI of an application in order to guide the user's initial experience along a happy path. These same methods could serve as a guided tour of a research prototype intended to showcase to reviewers novel contributions and design benefits, while bypassing unimplemented regions.

5.2 Philosophical reflections and scientific experimentalism

The fact that we now largely view programming as manipulation of linguistic entities is not because that would be the only way of programming, but rather, a consequence of historical developments that made programming language a *language* in the 1950s (Nofre et al., 2014). We believe that programming systems design research that we advocate in this paper needs to take a different approach. However, the ubiquity of the linguistic metaphor means that much of our academic infrastructure, including evaluation methods, is designed around it. The transformation of our research focus and of our research techniques can only proceed hand in hand. Interactive essays, as discussed in this section, are one of the aspects that can support this paradigm shift.

In the current paradigm, the central point of our knowledge is 'how'. Theoretical papers capture the formal essence of 'how', while empirical papers discuss practical implementation aspects of 'how'. If we adopt interactive essays as our way of disseminating results, we will be able to also build and grow knowledge focusing on 'what'. That is, what are the different user experience that a system based on a particular design can provide?

In philosophy of science, this attitude is akin to the experimentalist approach of Ian Hacking (1983). Hacking discusses problems with many accounts of scientific progress and proposes an alternative - science progresses by accumulating practical experimental knowledge. Developing a theory of electrons and positrons is not (yet) such knowledge, because our interpretation of what an electron or positron is might (and frequently does) change. However, learning how to change the charge of a niobium ball by spraying it with positrons or electrons a different kind of knowledge. Even if our theories change so that it does not even involve electrons and positrons as primitive entities, we can still conduct the practical experiment and new theories will have to find new explanations for it.

Similarly, a significant amount of programming systems knowledge is heavily reliant on the theory within which it was developed (an efficient way of compiling purely functional languages is not interesting to anyone using another kind of programming language). This should not be the case with

[3] For example, see the New York Times article by Almukhtar and Watkins (2016), where scrolling controls a timeline of an event illustrated using an animation on a map.

experiences documented in the form of interactive essay. Even if we start thinking about programming systems differently or use a different implementation method, we can still follow the documented user experience. We will either find a way of replicating it in a new system, or we will know that we are losing something valuable.

Acknowledgements

We thank for helpful comments: Jun Kato.

References

Almukhtar, S. and Watkins, D. (2016). How One of the Deadliest Hajj Accidents Unfolded. New York Times, Available online (retrieved 8 June 2019): https://www.nytimes.com/interactive/2016/09/06/world/middleeast/2015-hajj-stampede.html

Brusilovsky, P. (1994). Explanatory visualization in an educational programming environment: connecting examples with general knowledge. In International Conference on Human-Computer Interaction (pp. 202-212). Springer, Berlin, Heidelberg.

Carter, et al. (2016). Experiments in Handwriting with a Neural Network, Distill http://doi.org/10.23915/distill.00004

Green, T. R. G.; Petre, M. (1996). Usability analysis of visual programming environments: A 'cognitive dimensions' framework. Journal of Visual Languages and Computing. 7: 131–174.

Hacking, I. (1983). Representing and intervening: Introductory topics in the philosophy of natural science. Cambridge University Press

Grossman, T., Fanny Chevalier, and Rubaiat Habib Kazi (2016). Bringing research articles to life with animated figures. ACM Interactions 23, 4 (June 2016), 52-57. http://interactions.acm.org/archive/view/july-august-2016/bringing-research-articles-to-life-with-animated-figures

Hart, V.; Case, N. (2014) Parable of the Polygons: A Playable Post on the Shape of Society Available online at: https://ncase.me/polygons/ (retrieved 7 June 2019)

Hempel, B. (2019). Summary of Olsen's Evaluating User Interface Systems Research. https://people.cs.uchicago.edu/~brianhempel/Evaluating%20User%20Interface%20Systems%20Research%20-%20Graphical%20Summary.pdf

LIVE (2013) Proceedings of the 1st International Workshop on Live Programming. ICSE 2013

McLuhan, M.(1967). Medium is the massage. An inventory of effects. Penguin Books.

Nofre, D., Priestley, M., & Alberts, G. (2014). When technology became language: The origins of the linguistic conception of computer programming, 1950–1960. Technology and Culture, 55(1), 40-75.

Olsen, D. R. Jr. (2007). Evaluating User Interface Systems Research. UIST 2007.

Petricek, T (2016) Coeffects: Context-aware programming languages. http://tomasp.net/coeffects/

Plotkin, G. and Kahn, G. (1987). A structural approach to operational semantics. In proceedings of STACS'87, pp. 22-39. Aarhus University.

Polanyi, M. (2012). Personal knowledge. Routledge.

Seyser, D. and Zeiller, M. (2018). Scrollytelling–An Analysis of Visual Storytelling in Online Journalism. In 22nd International Conference Information Visualisation (IV). IEEE

Tanimoto, S. (1990) VIVA: A visual language for image processing. Journal of Visual Languages and Computing, Vol. 1, No. 2, June 1990.

UI-patterns (2019) Onboarding http://ui-patterns.com/patterns/onboarding/list retrieved June 2019

Victor, B. (2012). Learnable Programming. http://worrydream.com/LearnableProgramming/

Victor, B. (2011). Explorable Explanations. Available online at: http://worrydream.com/ExplorableExplanations/ (retrieved 7 June 2019)

Wootton, D. (2015). The Invention of Science: A New History of the Scientific Revolution. Allen Lane Books. ISBN 978-1846142109