HAL
open science

Towards Collaborative Prototyping Tools for Interaction Design

Germán Leiva, Nolwenn Maudet, Michel Beaudouin-Lafon

To cite this version:

Germán Leiva, Nolwenn Maudet, Michel Beaudouin-Lafon. Towards Collaborative Prototyping Tools for Interaction Design. Workshop: Digital Tools in Collaborative Creative Work, Sep 2018, Oslo, Norway. hal-03688610

HAL Id: hal-03688610

https://hal.science/hal-03688610v1

Submitted on 5 Jun 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Towards Collaborative Prototyping Tools for Interaction Design

Germán Leiva
Univ. Paris-Sud, CNRS,
Inria, Université Paris-Saclay
Orsay, France
leiva@lri.fr

Nolwenn Maudet
University of Tokyo
Tokyo, Japan
nolwenn@iis.u-tokyo.ac.jp

Michel Beaudouin-Lafon
Univ. Paris-Sud, CNRS,
Inria, Université Paris-Saclay
Orsay, France
mbl@lri.fr

ABSTRACT

Current prototyping tools poorly support the co-creation of novel interactions beyond standard widgets and touch-based interactions. We present two collaborative tools that let designers quickly prototype non-standard interactions. Montage supports an enhanced video prototyping process while Enact supports *design by enaction* through multiple interconnected viewpoints. We believe that such prototyping tools facilitate collaboration across the boundaries of communities of practice and enable the creation of truly evolutionary prototypes.

CCS CONCEPTS

• **Human-centered computing** → **Interface design prototyping**; **Systems and tools for interaction design**; *Collaborative and social computing*; • **Software and its engineering** → *Collaboration in software development*;

KEYWORDS

Designer-developer collaboration; Video prototyping; Interactive prototyping

INTRODUCTION

Cooper et al. define interaction design as "the practice of designing interactive digital products, environments, systems, and services" [9]. Interaction design borrows methods and techniques from other disciplines, such as product design and software engineering. Usually, these multiple responsibilities are distributed between designers and developers.

During the interaction design process, designers and developers articulate their work using various *design artifacts* such as sketches [7, 10], storyboards [19], and computational proxies [4]. These interaction design artifacts can generally be seen as *boundary objects* [24, 25]: shared artifacts used by different communities of practice to satisfy their information needs. The interpretive flexibility of boundary objects allows collaboration to proceed in the absence of consensus and standards.

We are interested in how these artifacts are collaboratively *co-created* [22], especially when designers and developers need to co-create non-standard interactive system, i.e. systems that include interactions that go beyond established design patterns. Within the past few years, over 40 commercial prototyping tools have emerged[1], and a 2015 survey of 4,000 designers found that 53% of them use such tools [27]. These commercial tools focus on supporting remote communication between design and development, assisting the extraction of design information for implementation, and helping to quickly prototype standard interactions.

Few tools, however, support the co-creation of the design artifacts, for example through collaborative graphical authoring software[2]. Usually, prototyping tools are used by a single community of practice [28], ignoring the back-and-forth between design and development. This results in focusing on a single viewpoint, either the designer's, e.g., visual representations, or the developer's, e.g., symbolic representations. Moreover, while current boundary objects such as sketches, storyboard, wireframes, and mock-ups excel at supporting the exploration of visual appearance, they poorly support the exploration of interaction behavior [18]. How can we provide better collaborative tools supporting boundary objects for the manipulation of dynamic and continuous interactions?

[1]http://www.cooper.com/prototyping-tools

[2]https://www.figma.com/

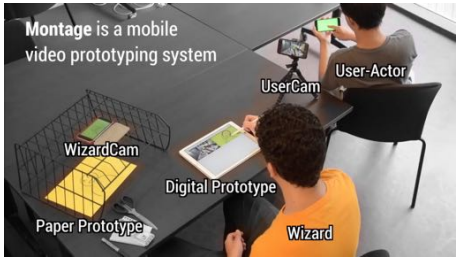

Figure 1: An overview of the Montage video prototyping system. Here, two designers, a wizard and a user-actor, collaborate side-by-side. The *UserCam* captures the user-actor enacting user inputs in the actual context, i.e. using his phone at the office. The *WizardCam* captures the paper prototype and the *Canvas* captures the wizard's digital sketches.

VIDEO AS A RICH MEDIUM FOR EXPLORING INTERACTION

Video prototyping [13–15] combines paper prototyping [23] with the Wizard-of-Oz technique [10] to create video artifacts that materialize the interaction design. Video artifacts can range from a simple and inexpensive recording of a traditional paper prototyping session [21] to a complex and expensive movie production [26]. Here, we are focusing on low-cost video prototypes in the early stages of the design. Such videos make it difficult to create dynamic transformations that re-shape or modify visual elements, such as re-sizing or stretching elements in response to continuous user input. Moreover, introducing changes in the prototype creates inconsistencies with previously recorded scenes, requiring re-shooting and/or post-production editing.

We created Montage [12], a new video prototyping tool to help a team of designers create video artifacts with minimal post-production editing and re-shooting. Montage is composed of a central device — the *Canvas* — connected to two mobile devices with video streaming and recording capabilities — the *UserCam* and the *WizardCam* (Figure 1). These devices, typically phones or tablets, are used as remote cameras, providing an inexpensive mobile movie studio. These cameras stream to the central *Canvas*, either in parallel or independently, the *context* of use in which the user interacts with the prototype and the prototyped *user interface* itself. The *Canvas* organizes the video clips into a grid of storylines. Montage lets designers compose shots using automatic green-screen replacement, whereby a video representing the interface replaces a green area of another video representing the use scenario [3]. Designers can also augment the interface or the scenario with digital sketches drawn

[3]See https://www.youtube.com/watch?v=7z_y8AX9gu8

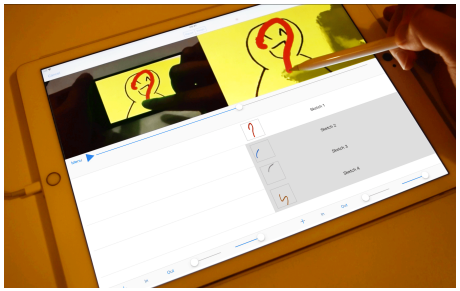

Figure 2: The *Canvas* sketching interface: The *final composition* (left) and the *interface* (right) show the "user overlay". Both sides have a list of sketches and animation controls at the bottom.

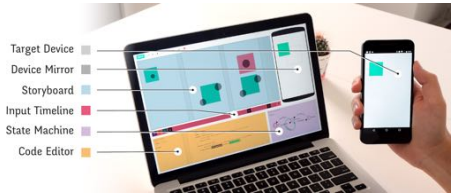

Figure 3: Enact uses a target mobile device and a desktop interface with five areas: a storyboard with consecutive screens, an event timeline with a handle for each screen, a state machine, a code editor and a device mirror

[4]See https://www.youtube.com/watch?v=uMJ9uNbPvwE

directly on the *Canvas*, without the need of re-shooting (Figure 2). Videos are backed up in a cloud server, allowing new remote cameras to join and expand the current video prototype with new scenes.

Montage provides multiple viewpoints that let designers collaborate simultaneously or independently. The *WizardCam* provides a physical prototyping area for paper prototyping. The *Canvas* provides a digital prototyping area for the creation of digital sketches. The *UserCam* provides an enactive [5] prototyping area where a user-actor can perform the interactions in context. The system is flexible enough to accommodate a variable number of designers. For example, the same person can act as paper prototyper and digital sketcher (Figure 1).

TOWARDS COLLABORATIVE INTERACTIVE PROTOTYPING

Video is a great medium to explore interactive behaviors within a given context of use, and systems such as Montage enable the collaborative co-creation of video artifacts. However, the final design artifact is a video prototype with no interactive capabilities for the viewer: the presenter can play, pause, rewind and fast-forward the video but the audience cannot interact with the prototype.

In order to create interactive prototypes of non-standard interactions, paper and video representations need to evolve into software-based representations. Research has produced a number of novel tools for creating interactive prototypes beyond coding, but these tools are targeted at a single community of practice rather than the collaborative work of designers and developers. Approaches such as programming by demonstration [17], state machines [1] or inference engines, for example, are interesting but have not been studied in a collaborative context.

After studying current designer-developer collaboration practices and challenges, we have identified three main collaborative *design breakdowns* [16]: *missing information*, when designers do not communicate specific details; *edge cases*, when designers do not think about particular cases; and *technical constraints*, when designers are not aware of technical limitations. These studies also showed that current workflows and tools induce unnecessary rework: Designers create a multitude of redundant design documents and developers must recreate them with their own tools, introducing mismatches with the original design. These findings show that designers still struggle to have a "conversation" with the software material [20].

We created Enact (Figure 3), a tool that reduces these design breakdowns during the collaborative prototyping of touch-based mobile interaction. Through multiple interconnected representations of the interaction under construction, Enact reduces reworking, redundancies and design breakdowns [4]. Enact features a linear storyboard that propagates changes from past to future screens, thus reducing redundancies within the storyboard. To facilitate navigation between visual and symbolic representations, the code editor is aware of the design elements in the storyboard: Visual elements can be dragged from the storyboard into the editor or directly accessed by their symbolic names, using intelligent code completion. Thanks to this tight integration, Enact supports visual representations

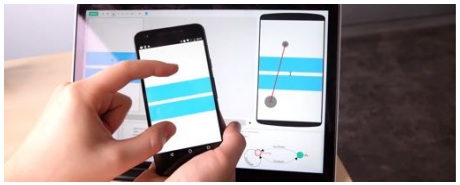

Figure 4: The designer created two measures, one between the touches and the other between the two rectangles. These measures are invisible on the target device (left) but they are revealed in the device mirror (right) and updated in real time to facilitate exploration.

Figure 5: On the left, the developer performs an off-device mimicking gesture with his left hand to understand the proposed design. On the right, the designer performs an on-device gesture with both hands to communicate the design.

for elements that are usually only available symbolically, such as *measures* that represent, e.g., the distance between two elements (Figure 4), or touch events representing the position of the finger and the radius of the contact area. Intermediate representations, such as the interactive state machine diagram, let designers and developers develop a common vocabulary, quickly explore the interaction under construction and detect edge cases.

The most important aspect of Enact is the use of the target device, in this case a mobile phone, not only as a testing device but also as a *design device*. The system relies on a similar approach to programming *with* examples [11] that we call *design by enaction*: Designers or developers can perform the desired user inputs directly on the target device as if they were the final user, in the same way as the user-actor in Montage; While acting out the interaction, Enact highlights the corresponding state in the state machine diagram and records every user input event. Designers can then define which screen of the storyboard is associated with which input event in the recorded sequence of user inputs. Developers can use this concrete example as a test to guide development [3]. Enact also provides an assisted testing feature that automatically replays the recorded example and highlights the mismatches between the implementation and the desired design.

CONCLUSION

Initiatives such as the Hour of Code [8] try to spread programming to a broader audience. Undoubtedly, more designers will know how to program in the future than today. However, code is not the only and most adequate representation for every aspect of interaction design. We need to provide multiple representations, e.g. symbolic, visual or enactive, to create collaborative spaces in which professionals with different skills, mindsets and values can work together. Digital tools need to provide ways of navigating these representations in a fluid way, thus reducing reworking and increasing reuse.

We outlined two collaborative prototyping tools for interaction design. Both projects emphasize sharing a common context, not only to prototype the interaction but also to prototype the user experience [6]. Montage encourages designers to create scenarios that can be explored with different alternatives of the system, or to assess a system in multiple scenarios. Enact supports *design by enaction*, where the target device is not only used for testing but also as a design medium to explore user interactions. While we focused on early stage prototyping, the ideas behind these tools can be applied to other collaborative tools. The ultimate goal is to better integrate such tools with the final digital product in order to create truly evolutionary prototypes [2].

ACKNOWLEDGMENTS

This work was partially supported by European Research Council (ERC) grants № 321135 “CREATIV: Creating Co-Adaptive Human-Computer Partnerships” and № 695464 “ONE: Unified Principles of Interaction”.

REFERENCES

[1] Caroline Appert and Michel Beaudouin-Lafon. 2006. SwingStates: adding state machines to the swing toolkit. In *Proceedings of the 19th annual ACM symposium on User interface software and technology - UIST '06*. ACM Press, New York, New York, USA, 319. http://dl.acm.org/citation.cfm?id=1166253.1166302

[2] Michel Beaudouin-Lafon and Wendy E. Mackay. 2003. Prototyping tools and techniques. In *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*. 1017–1039. https://doi.org/10.1201/9781410615862

[3] Kent Beck. 2003. *Test-driven development: by example*. Addison-Wesley Professional.

[4] Judith M. Brown, Gitte Lindgaard, and Robert Biddle. 2011. Collaborative Events and Shared Artefacts: Agile Interaction Designers and Developers Working Toward Common Aims. In *2011 AGILE Conference*. IEEE, 87–96. https://doi.org/10.1109/AGILE.2011.45

[5] Jerome S. Bruner. 1966. *Toward a theory of instruction*. Vol. 59. Harvard University Press.

[6] Marion Buchenau and Jane Fulton Suri. 2000. Experience prototyping. In *Proceedings of the conference on Designing interactive systems processes, practices, methods, and techniques - DIS '00*. ACM Press, New York, New York, USA, 424–433. https://doi.org/10.1145/347642.347802

[7] Bill Buxton. 2007. *Sketching User Experiences: getting the design right and the right design*. Morgan Kaufmann. 448 pages. https://doi.org/10.1016/B978-0-12-374037-3.X5043-3

[8] Code.org. 2018. Hour of Code. https://hourofcode.com/

[9] Alan Cooper, Robert Reimann, and David Cronin. 2007. *About face 3: the essentials of interaction design*. Vol. 3. John Wiley & Sons. 610 pages. https://doi.org/10.1057/palgrave.ivs.9500066

[10] Saul Greenberg, Carpendale Sheelagh, Marquardt Nicolai, and Buxton Bill. 2012. *Sketching User Experiences: The Workbook*. Morgan Kaufmann. 272 pages. https://doi.org/10.1016/C2009-0-61147-8

[11] Jun Kato, Takeo Igarashi, and Masataka Goto. 2016. Programming with Examples to Develop Data-Intensive User Interfaces. *Computer* 49, 7 (jul 2016), 34–42. https://doi.org/10.1109/MC.2016.217

[12] Germán Leiva and Michel Beaudouin-Lafon. 2018. Montage: A Video Prototyping System to Reduce Re-Shooting and Increase Re-Usability. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology - UIST '18*. ACM Press, Berlin, Germany. https://doi.org/10.1145/3242587.3242613

[13] Wendy E. Mackay. 1988. *Video Prototyping: a technique for developing hypermedia systems*. Vol. 5. ACM/SIGCHI.

[14] Wendy E. Mackay. 2002. Using video to support interaction design. *INRIA Multimedia Services* 2, 5 (2002). https://www.lri.fr/~mackay/VideoForDesign/

[15] Wendy E. Mackay and Anne-Laure Fayard. 1999. Video brainstorming and prototyping: techniques for participatory design. *CHI'99 extended abstracts on Human factors in …* May (1999), 118–119. https://doi.org/10.1145/632716.632790

[16] Nolwenn Maudet, Germán Leiva, Michel Beaudouin-Lafon, and Wendy E. Mackay. 2017. Design Breakdowns: Designer-Developer Gaps in Representing and Interpreting Interactive Systems. *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing - CSCW '17* (2017), 630–641. https://doi.org/10.1145/2998181.2998190

[17] Brad A. Myers, Richard G. McDaniel, and David Wolber. 2000. Programming by example: intelligence in demonstrational interfaces. *Commun. ACM* 43, 3 (mar 2000), 82–89. https://doi.org/10.1145/330534.330545

[18] Brad A. Myers, Sun Young Park, Yoko Nakano, Greg Mueller, and Andrew J. Ko. 2008. How designers design and program interactive behaviors. *Proceedings - 2008 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2008* (2008), 177–184. https://doi.org/10.1109/VLHCC.2008.4639081

[19] Mark W. Newman and James A. Landay. 2000. Sitemaps, Storyboards, and Specifications: A Sketch of Web Site Design Practice. In *Proceedings of the conference on Designing interactive systems processes, practices, methods, and techniques - DIS '00*. ACM Press, New York, New York, USA, 263–274. http://dl.acm.org/citation.cfm?id=347642.347758

[20] Fatih Kursat Ozenc, Miso Kim, John Zimmerman, Stephen Oney, and Brad A. Myers. 2010. How to support designers in getting hold of the immaterial material of software. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, New York, New York, USA, 2513. https://doi.org/10.1145/1753326.1753707

[21] Marc Rettig. 1994. Prototyping for tiny fingers. *Commun. ACM* 37, 4 (apr 1994), 21–27. https://doi.org/10.1145/175276.175288

[22] Elizabeth B.-N. Sanders and Pieter Jan Stappers. 2008. Co-creation and the new landscapes of design. *CoDesign* 4, 1 (mar 2008), 5–18. https://doi.org/10.1080/15710880701875068

[23] Carolyn Snyder. 2004. *Paper prototyping: The fast and easy way to design and refine user interfaces.* Morgan Kaufmann. 408 pages. https://doi.org/10.1016/B978-1-55860-870-2.X5023-2

[24] Susan Leigh Star. 1989. The Structure of Ill-structured Solutions: Boundary Objects and Heterogeneous Distributed Problem Solving. In *Distributed Artificial Intelligence (Vol. 2)*, M. Huhns (Ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 37–54. http://dl.acm.org/citation.cfm?id=94079.94081

[25] Susan Leigh Star and James R. Griesemer. 1989. Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social studies of science* 19, 3 (1989), 387–420. http://www.jstor.org/stable/285080

[26] Bruce Tognazzini. 1994. The "Starfire" video prototype project. In *Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94*. ACM Press, New York, New York, USA, 99–105. https://doi.org/10.1145/191666.191712

[27] Khoi Vinh. 2015. The Tools Designers Are Using Today. http://tools.subtraction.com/index.html

[28] Etienne Wenger. 1998. *Communities of practice: Learning, meaning, and identity.* Cambridge university press.