Towards Collaborative Prototyping Tools for Interaction Design

Germán Leiva, Nolwenn Maudet, Michel Beaudouin-Lafon

To cite this version:

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 systems, 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, storyboards, 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?

### 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\(^3\) 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. Designers can also augment the interface or the scenario with digital sketches drawn...

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. 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 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.

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

See https://www.youtube.com/watch?v=uMJ9uNbPvwE
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”.

---

**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.
REFERENCES


Towards Collaborative Prototyping Tools for Interaction Design


