Creating Interactive User Interfaces by Demonstration using Crowdsourcing

Rebecca Krosnick
Computer Science & Engineering | University of Michigan, Ann Arbor
rkros@umich.edu

I. Introduction

People are becoming increasingly interested in creating their own digital content and media. This is evident in the enormous number of blogs, personal websites, and portfolios available online. Website templates and creation/hosting services (e.g., Wix, WordPress, Google Sites) have made it possible for even non-programmers to create websites. However, with these services, non-programmers are limited to templates or basic user interface elements and behaviors, lacking the ability to create truly custom web pages that satisfy their needs. More complex and custom user interfaces like digital games and software are virtually impossible for non-programmers to create; even visual programming (e.g., Blockly, GameMaker Studio 2) and data flow languages that try to make computing more approachable still require an understanding of programming and computing concepts. As simple as it is for the average person to sketch a User Interface (UI) on paper or describe it in words, I believe it should be just as easy for them to create the actual digital UI with all of the desired behaviors. Programming should not be a barrier to creating new things and sharing them with the world.

Programming by Demonstration (PbD) has been an approach previously explored to enable end-user programmers to create programs without writing program code. End-users instead demonstrate how their program should work for example scenarios. There has been a rich body of work in PbD, with a number of papers focused on building interactive UIs and games [1] [2] [3]. Although these systems proved promising in lab studies, PbD has not seen much adoption in commercial products, a couple reasons being: 1) often many demonstrations are needed for the PbD system to correctly infer the end-user's intended behaviors, and 2) it can be difficult for end-users to understand what exact demonstrations are needed [4].

In my future work I plan to address these challenges in an effort to make PbD a more feasible approach for enabling end-user programmers to build custom, interactive UIs.

#1: Make it more feasible to gather many demonstrations: In prior PbD systems, it was assumed that the single end-user would create all demonstrations and answer the system's clarifying questions. This requires much effort from one person, and can make using such a system undesirable. I propose applying crowdsourcing to this problem, to spread the effort across multiple people. I am currently designing a crowdsourcing pipeline that leverages crowd workers to create PbD demonstrations capable of accurately satisfying the end-user's UI behavior requirements.

#2: Make it easier to gather the right demonstrations: To make it easier to gather demonstrations needed to correctly disambiguate and refine UI behaviors, I propose finding ways to guide crowd workers to create these demonstrations. I believe that by asking workers questions about what UI elements and properties a behavior is dependent on, the system can understand what new demonstration start-states or triggers would be informative and can ask workers to demonstrate the corresponding responses.

978-1-5386-4235-1/18/$31.00 ©2018 IEEE

II. PbD for Creating Dynamic UIs

As one step in the direction of enabling end-user programmers to build custom UIs, I have recently created Expresso [5], a PbD tool for building UIs with custom responsive behaviors, something that current template and website creation services do not support. Expresso does not require the user to write program code. With Expresso, a user starts with a static layout web page and then creates *keyframes* — examples of how the web page should look for different viewport widths — by directly manipulating UI elements in a WYSIWYG editor. Expresso can use a small set of keyframes to determine page layout for any viewport width. By default, the layout for a viewport width between two provided keyframes is the linear interpolation of the two keyframes' element property values, or a *smooth* transition. Expresso also supports discontinuous changes in layout as the viewport width is changed, for example a UI element being horizontally centered for small viewport widths and right-aligned for large viewport widths, which is enabled by the ability to set a *jump* transition between two keyframes. In a study I ran with participants who had minimal Cascading Style Sheets (CSS) experience [5], I saw that participants were able to build realistic responsive behaviors using Expresso. Although Expresso is effective for creating responsive UIs, it does not support creating more complicated behaviors, such as those in a digital game that may depend on interaction events and the state of various UI elements. Many more demonstrations would be needed to successfully encode such complicated behaviors using PbD. Using crowdsourcing could make creating a large number of demonstrations more manageable.

III. Related Work in Crowdsourcing

Crowdsourcing is the act of making an open call for people to complete work. It is often used to scale human computation, which integrates human intelligence into a computational process to complete work better than either humans or machines could alone. Recent work has shown that continuous real-time crowdsourcing [6] can be used for building and powering UI prototypes based on end-user requests. Apparition [7], [8] enables an end-user to use natural language and hand-sketches to communicate UI requirements. The crowd then implements a higher-fidelity prototype matching the requirements, and can Wizard-of-Oz animation requirements. SketchExpress [9] builds on Apparition by enabling workers to create, save, and reuse animation behaviors, which the end-user can replay later. However, neither of these systems support creating truly automated interactive UIs. At run time, a human must either manually animate behaviors (in Apparition) or manually press "play" buttons to replay recorded behaviors (in SketchExpress). Behaviors dependent on state changes or user interaction events are not supported. By instead leveraging crowd workers to create PbD demonstrations, a system can infer a UI behavior model that can applied to automatically render UI updates based on events and user interactions.

IV. Future Work

I am starting to design the crowdsourcing pipeline that will generate the PbD demonstrations necessary to define an end-user's requested UI. An end-user requester will first describe their UI behavior requirements by text or audio. Each description will then be sent to a crowd worker, who will be asked to create relevant demonstrations. Like some prior PbD systems, we will likely ask the worker to demonstrate a "UI before-state" and events (e.g., user interaction, timer event, UI change event), and then the resulting "UI after-state". A worker will likely need to provide multiple demonstrations for the UI to correctly exhibit the behavior they were assigned.

As with most crowdsourcing systems, the system should not blindly assume that any particular worker demonstration is correct. However, it would defeat the purpose to have the end-user check the validity of each worker demonstration; in that case, the end-user could have just spent their time creating all the demonstrations themselves. To address accuracy concerns while requiring zero or minimal work from the end-user, I plan to gather redundant demonstrations for the same ("before-state", event) pair from multiple workers. For a previously created demonstration, its ("before-state", event) could be passed to other workers, who would then be asked to demonstrate the expected "after-state". The system would then need some intelligent, and likely automated, scheme for aggregating the redundant demonstrations in order to generate the most accurate demonstration of the requested behavior as possible. Although asking for redundant demonstrations will increase the total amount of work required, I claim that the amount of work required for any single worker will still be less than an end-user providing demonstrations alone, as the system will not require every worker to complete every ("before-state", event) demonstration.

Since crowd workers will not be expert users of PbD or this system, it will be particularly important to make creating the right demonstrations easy. "Good" demonstrations are ones that are meaningfully diverse, demonstrating a wide range of the state-space and clarifying behavior differences for small state changes. Achieving such diversity will be helpful in building robust UIs that satisfy the end-user's requirements. It is likely that a worker may create a few initial demonstrations but then notice that the UI still does not completely satisfy the requester's behavior requirements. I hope to help workers create demonstrations for new, relevant ("before-state", event) pairs that would help the inference engine refine UI behaviors correctly. To do this, I intend to take prior ("before-state", event) pairs and perturb them in meaningful ways, in order to generate new ("before-state", event) pairs whose full demonstrations, completed by workers, would prove informative to the inference engine. To perturb ("before-state", event) pairs in meaningful ways, I plan to also ask workers questions about the semantics of the requested UI behaviors, for example whether an element's end location depends on its start location, or whether an element's color depends on another's.

Some other interesting questions to explore will be: What kinds of UI behavior descriptions can workers reliably understand, and which ones can they not? What kind of training will workers need to effectively use this system? Compared to an end-user performing all demonstrations themselves, how much faster can this crowdsourcing pipeline be?

I am excited about applying crowdsourcing to PbD as I think it could make PbD more feasible for end-user programmers. In general I am excited for a future where hopefully any person, regardless of technical expertise, can create custom programs and UIs that contribute to the world.

References

[1] B. A. Myers, "Peridot: creating user interfaces by demonstration," in *Watch what I do*. MIT Press, 1993, pp. 125–153.

[2] R. G. McDaniel and B. A. Myers, "Getting more out of programming-by-demonstration," in *Proc. of CHI*. ACM, 1999, pp. 442–449.

[3] M. R. Frank, P. N. Sukaviriya, and J. D. Foley, "Inference bear: designing interactive interfaces through before and after snapshots," in *Proc. of DIS*. ACM, 1995, pp. 167–175.

[4] B. A. Myers and T. J. J. Li, "Teaching intelligent agents new tricks: Natural language instructions plus programming-by-demonstration for teaching tasks." Human Computer Interaction Consortium (HCIC), 2018.

[5] R. Krosnick, S. W. Lee, W. S. Lasecki, and S. Oney, "Expresso: Building responsive interfaces with keyframes," in *Proc. of VL/HCC*. IEEE, 2018.

[6] W. S. Lasecki, K. I. Murray, S. White, R. C. Miller, and J. P. Bigham, "Real-time crowd control of existing interfaces," in *Proc. of UIST*. ACM, 2011, pp. 23–32.

[7] W. S. Lasecki, J. Kim, N. Rafter, O. Sen, J. P. Bigham, and M. S. Bernstein, "Apparition: Crowdsourced user interfaces that come to life as you sketch them," in *Proc. of CHI*. ACM, 2015, pp. 1925–1934.

[8] S. W. Lee, R. Krosnick, B. Keelean, S. Vaidya, S. D. O'Keefe, S. Y. Park, and W. S. Lasecki, "Exploring real-time collaboration in crowd-powered systems through a ui design tool," in *Proceedings of the ACM Conference on Computer-Supported Cooperative Work and Social Computing*. ACM, 2018.

[9] S. W. Lee, Y. Zhang, I. Wong, Y. Y., S. O'Keefe, and W. Lasecki, "Sketchexpress: Remixing animations for more effective crowd-powered prototyping of interactive interfaces," in *Proc. of UIST*. ACM, 2017.