Code and Contribution in Interactive Systems Research

James Fogarty
Computer Science & Engineering
DUB Group | University of Washington
jfogarty@cs.washington.edu

ABSTRACT

The scale and complexity of interactive systems research often require care in distinguishing: (1) the code that implements a system, versus (2) the research contribution demonstrated or embodied in a system. This position paper for the CHI 2017 workshop on #HCI.Tools reflects on this contrast and some common forms of contribution in interactive systems research. We explore several forms of interactive systems contribution based in differentiating: (1) *what* a system accomplishes, versus (2) *how* it accomplishes that. We argue some interactive systems should be considered sketches that use code as a medium to explore their research contributions, while others embody their contributions more directly in their code. Finally, we argue the progress and impact of our field requires diverse forms of contribution across interactive systems.

INTRODUCTION

The scale and complexity of modern interactive systems is daunting along several dimensions. Weiser characterized important aspects of this in a trend from many-to-1 (i.e., many people sharing a single device), to 1-to-1 (i.e., each person with a dedicated device), to 1-to-many (i.e., each person having many devices), to many-to-many (i.e., many people connected through many devices) [14]. As technology enters later stages of this trend, researchers now explore interactive systems that span multiple devices, require massive volumes of data to enable seemingly simple interactions, or require entire social networks before key aspects of their design can surface. Such barriers to real-world deployment of interactive systems create important challenges for interactive systems research.

This reflection focuses primarily on the relationship between code and contribution. Interactive systems research generally contains both, but they are not always well-distinguished. Prior discussions include consideration of the limitations of usability testing [6], examination of common pitfalls in evaluating interactive systems [11], and discussion of technical HCI research as an activity of invention that contrasts with activities of discovery [8]. Additional discussion considers how these challenges manifest or can be magnified in social computing systems [1], with their corresponding need for a critical mass of participation [7]. Our reflection is intended to complement existing discussions without contradiction.

Presented at the CHI 2017 Workshop on #HCI.Tools: Strategies and Best Practices for Designing, Evaluating, and Sharing Technical HCI Toolkits.

This position paper first considers the case where code is closely linked to contribution. It then explores cases where the link is less direct. Consistent with the workshop's proposal to explore conceptual roles for toolkits in HCI research, we examine several forms of interactive systems contribution based in a differentiation of: (1) *what* a system accomplishes, versus (2) *how* it accomplishes that. We conclude with brief comments on our prior interactive systems research as a background for participation in the #HCI.Tools workshop.

WHEN CODE IS THE CONTRIBUTION

Some interactive systems research contributions are directly manifested in code. Although these are a minority, they are important for both: (1) their own research value and impact, and (2) the contrast they can provide for other styles of research. A well-known example is the $1 Recognizer, a template-based unistroke gesture recognizer implemented in approximately 100 lines of code [15]. The paper has been widely cited, both in applications that use the recognizer and in later extensions of the underlying recognition technique. A project website also hosts community implementations of the recognizer in multiple programming languages. The contribution and impact of this research thus directly results from solving a technical challenge in code that people can easily adopt and adapt in their applications and contexts.

Replication, Validation, and Extension

Discussions of replication within the CHI community often focus on experimental replication, which remains relevant in our current context. For example, the $1 Recognizer's project website includes data to replicate its performance experiments. But contributions associated with code also provide opportunity for stronger validation: each future use of that code in a new application, or in a context beyond the original research, validates the underlying research contribution. This validation is riskier and therefore stronger than simply re-executing the original data analysis or replicating the prior experiment.

Figures 1 and 2 illustrate this using a simple visual language we develop in figures throughout this paper. In Figure 1, we distill the contribution of the $1 Recognizer down to a circle. The circle is filled (i.e., purple) to indicate that contribution is novel. In contrast, we will use empty circles (i.e., white) to illustrate components of a system that are not themselves novel (e.g., replicate a prior result, otherwise already known). Figure 2 illustrates this in a research progression based on the $1 Recognizer. This progression begins with Protractor, a recognizer informed by techniques in the $1 Recognizer [9]. Protractor is then used in implementing Gesture Script, a novel tool for interactively authoring compound gestures [10].

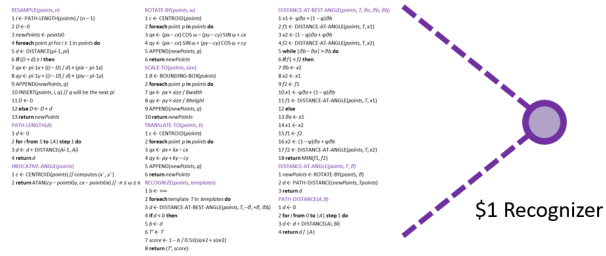

Figure 1: The $1 Recognizer's contribution closely corresponds to its code, allowing re-use of its solution to a technical problem.

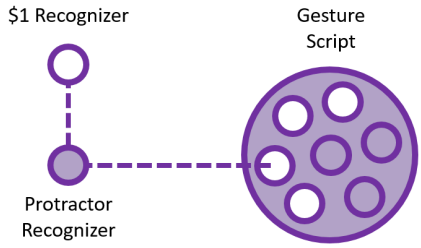

Figure 2: Protractor extends techniques introduced in the $1 Recognizer. It is therefore novel (i.e., filled), while also replicating and extending the $1 Recognizer (i.e., now shown as empty). Gesture Script then directly replicates Protractor as part of a larger system that also includes other components. Some of those are already known (i.e., are also empty), while some are novel (i.e., filled). The overall functionality presented by Gesture Script is also novel (i.e., the large circle is filled).

At each step in this progression, previously novel elements (i.e., filled circles) become already known (e.g., the circle illustrating Gesture Script's use of Protractor is now empty). Gesture Script is itself novel (i.e., the large circle is filled), uses several known techniques (i.e., empty inner circles), and requires several underlying innovations (i.e., filled inner circles). Many other progressions similarly trace back to the $1 Recognizer, and such illustrations can be considered at multiple scales. If we were to "zoom in" on Protractor relative to the $1 Recognizer, we would see they have many identical inner circles, but Protractor includes a novel closed-form similarity metric (i.e., a novel inner circle).

WHEN CODE IS NOT THE CONTRIBUTION

Most interactive systems contributions do not correspond so directly to their code. Identifying the contributions of a system can therefore be considered in terms of: (1) *what* a system accomplishes (i.e., the outer circle and its novelty), and (2) *how* a system accomplishes that (i.e., the inner circles and their novelty). We consider three important combinations.

Figure 3 illustrates the easiest case, for which interactive systems researchers often strive, and for which interactive systems reviewers are often most comfortable. The illustrated system accomplishes novel functionality (i.e., the outer circle, the *what*), enabled in part by novel techniques (i.e., the inner circles, the *how*). Such contributions can often be motivated either bottom-up (i.e., introducing novel inner techniques and then novel outer functionality they enable) or top-down

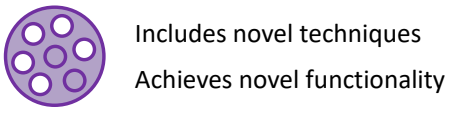

Figure 3: Interactive systems that include both novel functionality (i.e., the outer *what*) and novel techniques (i.e., the inner *how*) can often be motivated and validated in either contribution.

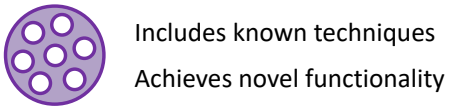

Figure 4: When underlying techniques are known (i.e., the inner *how*), the question is whether their combination in new functionality is a significant contribution (i.e., the outer *what*).

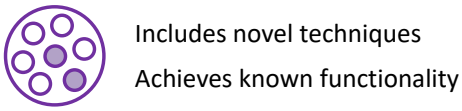

Figure 5: When applied in known overall functionality (i.e., the outer *what*), the question is whether implications of novel inner techniques are a significant contribution (i.e., the inner *how*).

(i.e., introducing novel outer functionality and then novel inner techniques required to achieve that functionality). At the scale of a paper, validation can often focus on either the outer circle (i.e., the *what*) or the inner (i.e., the *how*). Researchers therefore have flexibility in deciding what to highlight. Individual reviewers may prefer novelty with regard to either the *what* (i.e., the outer circle) or the *how* (i.e., the inner circles), but a system that makes novel contributions in both is more robust to such variation in reviewer taste.

Figure 4 illustrates a system that uses known techniques to accomplish novel functionality. The challenge for researchers and reviewers is deciding whether the outer circle (i.e., the *what*) is a significant contribution (i.e., versus the case where both outer and inner circles are already known). Reasonable people may disagree regarding significance of contribution in the outer circle, but it seems inappropriate to dismiss any contribution because "we already know" the inner circles. Such work is sometimes dismissed as "obvious", but if no prior work has previously combined these pieces into the resulting functionality, a more appropriate question is whether that novel functionality is a significant contribution.

Figure 5 conversely illustrates a system that applies novel techniques in the context of known overall functionality. The challenge is deciding whether improvements to the inner circles are a significant contribution, and it seems inappropriate to dismiss that potential because the outer circle "has been done". Such work is sometimes dismissed as "just engineering", but a more appropriate question is whether the implications of novel inner techniques make them a significant contribution. These can include advances in performance or robustness, but a researcher is often wise to show how advances impact functionality (i.e., to convert this into our first case by showing how inner contributions enable novelty in the outer circle).

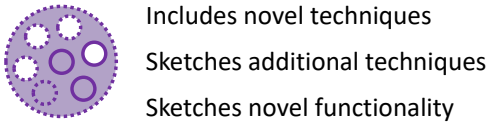

Figure 6: Many research systems are sketches supporting a research contribution. This example includes concrete and novel contributions (i.e., solid inner filled circles) as part of a larger system sketching novel functionality. Dashed elements need additional work before they are fully achieved, but the sketch allows critique to focus on the current contributions.

Scale and Sketching

Given the above forms of contribution, we now return to the problem of scale and complexity. If a researcher's intended contribution is the outer circle (i.e., the *what*), then elements of inner circles may be irrelevant. For example, consider a system that requires persistent storage, but has no interesting requirements of that storage. A decision to use a local file, a local database, or a cloud database will impact the system's code, but is irrelevant to its research contribution. Conversely, if the intended contribution is an inner circle (i.e., the *how*), details of an outer circle may be irrelevant. Overall, a researcher is generally not developing a product and will make choices that impact code according to whatever is most expedient without sacrificing the research contribution. Instead of criticizing this as "research code", or demanding unreasonable standards, we must remember the researcher pursues different goals than a product developer.

We believe many interactive systems developed in research should be considered sketches, as described by Buxton [2]. Sketches allow rapid exploration of many possibilities, with each sketch surfacing its key properties for critique. Sketches are also intentionally left ambiguous in many ways, with additional details to be defined if the idea is further pursued. This property expedites sketching because it allows proceeding without spending time or resources defining details. It also improves critique by remaining focused on important aspects.

Figure 6 extends prior examples to illustrate this, using dashed circles to show sketched elements. The system includes novel and concrete techniques, but other elements remain sketched. The system works well enough to demonstrate the proposed functionality and to validate the novel techniques that were developed. But fully achieving its proposed functionality still requires additional work implementing known techniques (i.e., dashed inner empty circles) and additional research addressing remaining challenges (i.e., dashed inner filled circles). Most research systems are sketches in this regard, emphasizing key contributions while leaving other aspects underdeveloped. Many demonstrations are also sketches, aiming to validate the contribution of an inner circle by sketching multiple outer circles that are potentially enabled.

Visions and Realizations

Even more than a sketch, a technology vision suggests a direction while leaving many unanswered questions in how such a vision will actually be achieved. Figure 7 illustrates this with larger holes in the vision. Realizing the vision thus

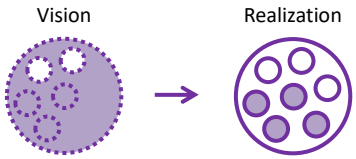

Figure 7: Research visions leave larger holes, in the form of more questions that need to be answered to realize the vision.

requires both: (1) research that addresses known challenges (i.e., circles that were sketched in the vision), and (2) research that defines and then addresses the larger holes in the vision.

Although the difference between a sketch and a vision is obvious at the extremes, the boundary between them is unclear and likely based in a judgment regarding the size of the holes. In visions with larger holes, it is increasingly likely the outer circle (i.e., the *what*, the vision) will be significantly changed in its realization (e.g., by implications of the specific inner circles developed in pursuing the vision), and therefore should often be considered a novel contribution. But even when a realization remains close to an original vision, there are often contributions in the inner circles needed to achieve that vision.

DISCUSSION

We have aimed to unpack common forms of contribution in interactive systems research, arguing those contributions are: (1) distinct from each other, and (2) often distinct from the code that is used as a medium to demonstrate or manifest a contribution. Our reflection is motivated by challenges we observe in researchers and reviewers conflating these aspects of work. If researchers believe their contributions are in one regard, while reviewers consider them in another, resulting mutual frustration generally undermines progress in our field.

In focusing on differentiating the *what* from the *how*, we have intentionally not engaged questions of validating either form of contribution. We have also not engaged questions of how much sketching is acceptable in a system, versus what aspects of a system must be more complete to be considered a contribution. Such questions seem better addressed in more specific contexts where they can be grounded in details of the work, but several points can be discussed more generally.

Irrelevant Detail and Irrelevant Replication

Some common pitfalls emerge when: (1) the difference between code and contribution is confused, or (2) notions of replicability in experimental contexts are misapplied in interactive systems. Consider the sketch from Figure 6, with a pair of novel techniques (i.e., solid filled inner circles). These techniques are intended as contributions and should be thoroughly reported so they can be understood and considered by reviewers. But desire for thoroughness sometimes leads reviewers to probe irrelevant details of a sketch. A known technique (i.e., a solid empty inner circle) has previously been validated. Use of a known technique is further validation, and it is likely inappropriate to expect the current work to explicitly revisit its validation. Similarly, the sketched inner circles should be considered only to the extent they impact the intended contribution. The choice of

how to implement these techniques, in the current sketch and in any future realizations, obviously impacts the code of such systems. But probing at unresolved details of these sketches, or attempting to evaluate such irrelevant details, is often obscuring the work's actual contribution.

Promoting Diverse Forms of Contribution

The long-term health and impact of our field requires all of the forms of contribution considered here. Visions can inspire other researchers to pursue a direction, allowing the field to explore and understand that space more quickly and effectively than waiting for the original researcher to "fill in the holes". Research systems that sketch relationships between *what* (i.e., their outer circles) and *how* (i.e., their inner circles) similarly allow the field to better explore and understand such relationships without them being hindered or obscured by other irrelevant details. But visions and sketching have limits. Achieving a full realization may reveal that prior sketches were incomplete or incorrect in important aspects of an idea. Full realizations also allow confident incorporation of prior work in new explorations, a contrast to stacking sketches that may eventually crumble under their own incompleteness. Full realizations thus enable both direct impact of the current research and future exploration of additional research.

From this perspective, it seems strange and unfortunate for our field to simultaneously lament: (1) a perception among researchers that innovation and novelty are limited by questions of validation that seem to work against exploring new directions, and (2) a perception among researchers that progress is limited by novelty fetishes that seem to work against building upon what is already known in pursuing deeper understanding and impact. We obviously need both, need authors and reviewers to be clear which is pursued, and need discussions of contribution and validation to be based in how specific work contributes to this balance.

PRIOR INTERACTIVE SYSTEMS RESEARCH

We look forward to workshop discussions of these and other perspectives on interactive systems. As background, our prior interactive systems research includes toolkits for sensor-based statistical models [5], exploration of tool challenges applying machine learning in everyday applications [12], techniques enabling graphical interfaces composed of mutually untrusted elements [13], a gesture authoring tool [10], and techniques and tools enabling pixel-based interpretation and runtime modification of graphical interfaces (e.g., [3,4,16]).

ACKNOWLEDGMENTS

This reflection began as a discussant talk in the HCIC 2011 workshop. We thank Sean Munson and Jacob O. Wobbrock for their persistent encouragement to prepare a written version of this reflection. This work was supported in part by the National Science Foundation under awards IIS-1053868 and SCH-1344613.

REFERENCES

1. Michael S. Bernstein, Mark S. Ackerman, Ed H. Chi, and Robert C. Miller. (2011). The Trouble with Social Computing Systems Research. *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems (CHI 2011)*, 389–398.
2. Bill Buxton. (2007). *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann.
3. Morgan Dixon and James Fogarty. (2010). Prefab: Implementing Advanced Behaviors Using Pixel-Based Reverse Engineering of Interface Structure. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2010)*, 1525–1534.
4. Morgan Dixon, Gierad Laput, and James Fogarty. (2014). Pixel-Based Methods for Widget State and Style in a Runtime Implementation of Sliding Widgets. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2014)*, 2231–2240.
5. James Fogarty and Scott E. Hudson. (2007). Toolkit Support for Developing and Deploying Sensor-Based Statistical Models of Human Situations. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2007)*, 135–144.
6. Saul Greenberg and Bill Buxton. (2008). Usability Evaluation Considered Harmful (Some of the Time). *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2008)*, 111–120.
7. Jonathan Grudin. (1988). Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces. *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW 1988)*, 85–93.
8. Scott E. Hudson and Jennifer Mankoff. (2014). Concepts, Values, and Methods for Technical Human-Computer Interaction Research. In *Ways of Knowing in HCI*. Springer, 69–93.
9. Yang Li. (2010). Protractor: A Fast and Accurate Gesture Recognizer. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2010)*, 2169–2172.
10. Hao Lü, James Fogarty, and Yang Li. (2014). Gesture Script: Recognizing Gestures and their Structure Using Rendering Scripts and Interactively Trained Parts. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2014)*, 1685–1694.
11. Dan R. Olsen, Jr. (2007). Evaluating User Interface Systems Research. *ACM Symposium on User Interface Software and Technology (UIST 2007)*, 251–258.
12. Kayur Patel, Naomi Bancroft, Steven M. Drucker, James Fogarty, Amy J. Ko, and James A. Landay. (2010). Gestalt: Integrated Support for Implementation and Analysis in Machine Learning. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2010)*, 37–46.
13. Franziska Roesner, James Fogarty, and Tadayoshi Kohno. (2012). User Interface Toolkit Mechanisms for Securing Interface Elements. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2012)*, 239–250.
14. Mark Weiser and John Seely Brown. (1996). The Coming Age of Calm Technology. Xerox PARC.
15. Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. (2007). Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2007)*, 159–168.
16. Xiaoyi Zhang, Anne Ross, James Fogarty, Anat Caspi, and Jacob O. Wobbrock. (2017). Interaction Proxies for Runtime Repair and Enhancement of Mobile Application Accessibility. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2017)*, To Appear.