Evaluation Strategies for HCI Toolkit Research

David Ledo[1*], Steven Houben[2*], Jo Vermeulen[3*], Nicolai Marquardt[4], Lora Oehlberg[1] and Saul Greenberg[1]

[1] Department of Computer Science, University of Calgary, Canada · {first.last}@ucalgary.ca
[2] School of Computing and Communications, Lancaster University, UK · s.houben@lancaster.ac.uk
[3] Department of Computer Science, Aarhus University, Denmark · jo.vermeulen@cs.au.dk
[4] UCL Interaction Centre, University College London, UK · n.marquardt@ucl.ac.uk
[*] *Authors contributed equally to the work.*

ABSTRACT

Toolkit research plays an important role in the field of HCI, as it can heavily influence both the design and implementation of interactive systems. For publication, the HCI community typically expects toolkit research to include an evaluation component. The problem is that toolkit evaluation is challenging, as it is often unclear what 'evaluating' a toolkit means and what methods are appropriate. To address this problem, we analyzed 68 published toolkit papers. From our analysis, we provide an overview of, reflection on, and discussion of evaluation methods for toolkit contributions. We identify and discuss the value of four toolkit evaluation strategies, including the associated techniques that each employs. We offer a categorization of evaluation strategies for toolkit researchers, along with a discussion of the value, potential limitations, and trade-offs associated with each strategy.

Author Keywords

Toolkits; user interfaces; prototyping; design; evaluation.

ACM Classification Keywords

H.5.2. Information interfaces and presentation (e.g. HCI): User Interfaces – Evaluation/methodology.

INTRODUCTION

Within HCI, Greenberg [30] defined toolkits as a way to encapsulate interface design concepts for programmers, including widget sets, interface builders, and development environments. Such toolkits are used by designers and developers to create interactive applications. Thus, they are generative platforms designed to create new artifacts, while simplifying the authoring process and enabling creative exploration.

While toolkits in HCI research are widespread, researchers experience toolkit papers as being hard to publish [77] for various reasons. For example, toolkits are sometimes considered as merely engineering, as opposed to research, when in reality some interactive systems are 'sketches' using code as a medium to explore research contributions, whereas others embody their contributions in the code itself [27]. Sometimes, toolkit researchers are asked for a particular evaluation method without consideration of whether such an evaluation is necessary or appropriate to the particular toolkit contribution. Consequently, acceptance of toolkits as a research contribution remains a challenge and a topic of much recurrent discussion [8,27,30,66,73,82]. In line with other areas of HCI [30,82], we should expect HCI toolkit research to use appropriate evaluation methods to best match the particular research problem under consideration [31,45,86]. However, while research to date has used evaluation methods, there is little overall reflection on *what* methods are used to evaluate toolkits, *when* these are appropriate, and *how* the methods achieve this through different techniques.

The last two decades have seen an increase in HCI toolkit papers [66]. These papers typically employ a range of evaluation methods, often borrowing and combining techniques from software engineering, design, and usability evaluation. From this corpus, we can consider how toolkit researchers collectively derive *what* evaluation methods are useful, *when* they are appropriate and *how* they are performed.

Based on an analysis of 68 representative toolkit papers, this paper contributes an overview and in-depth discussion of evaluation methods for toolkits in HCI research. We identify four types of evaluation strategies: (1) *demonstration*, (2) *usage*, (3) *technical benchmarks*, and (4) *heuristics*. We present these four evaluation types, and opine on the value and limitations associated with each strategy. Our synthesis is based on the sample of representative toolkit papers. We link interpretations to both our own experiences and earlier work by other toolkit researchers. Researchers can use this synthesis of methods to consider and select appropriate evaluation techniques for their toolkit research.

WHAT IS A TOOLKIT?

Within HCI literature, the term 'toolkit' is widely used to describe various types of software, hardware, design and conceptual frameworks. Toolkit research falls into a category of *constructive research,* which Oulasvirta and Hornbæk define as "*producing understanding about the construction of an interactive artefact for some purpose in human use of computing*" [83]. They specify that constructive research is driven by the absence of a (full) known solution or resources to implement and deploy that solution.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

ACM ISBN 978-1-4503-5620-6/18/04….$15.00
https://doi.org/10.1145/3173574.3173610

#	TOOLKIT	VENUE	YEAR	REF	TYPE 1	TYPE 2	TYPE 3	TYPE 4
1	Context Toolkit	CHI	1999	[91]	■			■
2	DENIM	CHI	2000	[60]	■	■		
3	Toolkit-Level Support for Ambiguity in Recognition-Based Interfaces	CHI	2000	[62]	■			
4	SATIN	UIST	2001	[41]	■		■	
5	Phidgets	UIST	2002	[32]	■	■		
6	Speakeasy	UIST	2003	[80]	■			
7	iStuff	CHI	2003	[4]	■			
8	MAUI Groupware Toolkit	CSCW	2004	[40]	■	■	■	
9	DiamondSpin	CHI	2004	[98]	■			
10	Papier-Mâché	CHI	2004	[52]	■	■	■	
11	Calder	DIS	2004	[57]	■		■	
12	ICON	ICMI	2004	[22]	■			
13	Toolkit Design for Interactive Structured Graphics	TSE	2004	[6]		■	■	
14	DART	UIST	2004	[61]	■	■		
15	MaggLite Post-WIMP Toolkit	UIST	2004	[47]	■		■	
16	Peripheral Displays Toolkit	UIST	2004	[68]	■			
17	Prefuse	CHI	2005	[38]	■	■		
18	SubArctic	CHI	2005	[46]	■			
19	d.Tools	UIST	2006	[37]	■	■		
20	SwingStates	UIST	2006	[2]	■	■		
21	Exemplar	CHI	2007	[36]		■		■
22	CapToolkit	PerCom	2007	[109]	■		■	
23	ReacTIVision	TEI	2007	[48]	■			
24	Shared Phidgets	TEI	2007	[65]	■	■		
25	VoodooIO	TEI	2007	[101]	■			
26	Gummy	AVI	2008	[71]	■	■		■
27	Damask	CHI	2008	[59]	■	■		
28	VoodooSketch	TEI	2008	[12]	■			
29	GT/SD	EICS	2009	[23]	■			
30	PyMT	ITS/ISS	2009	[35]	■			
31	Protovis	TCVG	2009	[13]	■	■	■	
32	Sikuli	UIST	2009	[115]	■	■	■	■
33	Intuino	DIS	2010	[103]	■	■		
34	Amarino Toolkit	MobileHCI	2010	[50]	■	■		
35	Intelligibility Toolkit	Ubicomp	2010	[58]	■			■
36	D-MACS	UIST	2019	[70]	■			■
37	Prefab	CHI	2010	[20]	■			
38	TouchID Toolkit	ITS/ISS	2011	[67]	■			
39	D3	TCVG	2011	[14]	■		■	
40	Proximity Toolkit	UIST	2011	[64]	■	■		
41	Phybots	DIS	2012	[49]	■	■		
42	jQMultiTouch	EICS	2012	[76]	■	■		
43	PuReWidgets	EICS	2012	[15]	■			
44	.NET Gadgeteer	Pervasive	2012	[102]	■	■		
45	HapticTouch Toolkit	TEI	2012	[55]	■	■		
46	Midas	UIST	2012	[95]	■	■		
47	PaperBox	C&C	2012	[107]	■	■		
48	WorldKit	CHI	2012	[113]	■			
49	KinectArms	CSCW	2013	[28]	■	■	■	
50	OpenCapSense	PerCom	2013	[33]	■		■	
51	ToyVision Toolkit	TEI	2013	[63]	■			
52	Sauron	UIST	2013	[93]	■	■		
53	PanelRama	CHI	2014	[114]	■	■		
54	XDStudio	CHI	2014	[75]	■	■	■	
55	XDKinect	EICS	2014	[74]	■	■		■
56	PolyChrome	ITS/ISS	2014	[3]	■		■	
57	PaperPulse	CHI	2015	[88]	■	■		
58	WatchConnect	CHI	2015	[43]	■			■
59	Weave	CHI	2015	[17]	■	■	■	
60	SoD Toolkit	ITS/ISS	2015	[96]	■			■
61	C4	TEI	2015	[51]	■	■	■	
62	Makers' Marks	UIST	2015	[94]	■			
63	Physikit	CHI	2016	[42]		■		
64	Retrofab	CHI	2016	[87]	■	■		
65	Let Your Body Move Toolkit	MobileHCI	2016	[84]	■	■		
66	CircuitStack	UIST	2016	[104]	■	■	■	
67	EagleSense	CHI	2017	[112]	■		■	
68	Pineal	CHI	2017	[54]	■			

Table 1. Overview of all toolkits in the sample. Types: (1) Demonstration, (2) Usage, (3) Technical Performance and (4) Heuristics.

As constructive research, toolkits examine new conceptual, design or technical solutions to unsolved problems. To clarify our review's scope, we define and summarize what is meant by "toolkit" and "toolkit evaluation", and why HCI researchers build toolkits.

Defining a Toolkit

We extend Greenberg's original definition [30] to define toolkits as *generative platforms designed to create new interactive artifacts, provide easy access to complex algorithms, enable fast prototyping of software and hardware interfaces, and/or enable creative exploration of design spaces*. Hence, toolkits present users with a programming or configuration environment consisting of many defined permutable building blocks, structures, or primitives, with a sequencing of logical or design flow affording a *path of least resistance*. Toolkits may include automation (e.g. recognizing and saving gestures [67]) or monitoring real-time data (e.g. visualization tools [52,64]) to provide developers with information about their own process and results.

Why Do HCI Researchers Build Toolkits?

Before discussing toolkit evaluation, we elaborate on what they contribute to HCI research. Wobbrock and Kientz position toolkits as *artifact* contributions, where *"new knowledge is embedded in and manifested by artifacts and the supporting materials that describe them"* [111]. We summarize discussions by Myers et. al [73], Olsen [82] and Greenberg [30] on the *value* of HCI toolkits into five goals:

G1. Reducing Authoring Time and Complexity. Toolkits make it easier for users to author new interactive systems by encapsulating concepts to simplify expertise [30,82].

G2. Creating Paths of Least Resistance. Toolkits define rules or pathways for users to create new solutions, leading them to right solutions and away from wrong ones [73].

G3. Empowering New Audiences. Given that toolkits reduce the effort to build new interactive solutions, they can enable new audiences to author these solutions. For example, Olsen [82] discusses how interface builders opened interface design to artists and designers.

G4. Integrating with Current Practices and Infrastructures. Toolkits can align their ideas to existing infrastructure and standards, enabling power in combination [82] and highlighting the value of infrastructure research for HCI [25]. For example, D3 [14] integrated with popular existing standards, which arguably contributed significantly to its uptake.

G5. Enabling Replication and Creative Exploration. Toolkits allow for replication of ideas that explore a concept [30], which collectively can create a new suite of tools that work together to enable scale and create *"larger and more powerful solutions than ever before"* [82].

Evaluating Toolkits

A common concern among HCI toolkit and system researchers is the difficulty in publishing [77]. This might be due to the expectations and prevalence of evaluation methods (e.g. user studies), regardless of whether the methods are necessary or appropriate to the toolkit's contribution. Part of the problem is a lack of clear methods [77] or a clear definition of 'evaluation' within a toolkit context. As toolkit designers, our stance is that the evaluation of a toolkit *must* stem from the claims of the paper. Evaluation is a means to follow through with the proposed claims of the innovation. We should ask ourselves: *what do we get out of the evaluation?*

Toolkits are typically different from systems that perform one task (e.g. a system, algorithm, or an interaction technique) as they provide generative, open-ended authoring within a solution space. Toolkit users can create different solutions by reusing, combining and adapting the building blocks provided by the toolkit. Consequently, the trade-off to such generative power is the large space that remains under explored. Evaluation methods that only examine a small subset of the toolkit may not demonstrate the research contribution, nor do they necessarily determine a toolkit's success. As summarized by Olsen [82] in his reflective paper on evaluat-

ing systems research: "*simple metrics can produce simplistic progress that is not necessarily meaningful.*" The central question is thus: *what is an evaluation?* And, *how do we reflect and evaluate such complex toolkit research?*

METHODOLOGY

This paper elucidates evaluation practices observed in modern toolkit research within the HCI community. To build up an in-depth understanding of contemporary evaluation practices, we report the results of a meta-review based on an analysis of a representative set of toolkit papers.

Dataset

To collect a representative set of HCI toolkit papers, we gathered 68 papers matching the following inclusion criteria.

Publication Venue and Date, Keywords: we initially selected 58 toolkit papers that were published since 2000 at the major ACM SIGCHI venues (CHI, UIST, DIS, Ubicomp, TEI, MobileHCI). We included papers containing keywords: *toolkit, design tool, prototyping tool, framework, API*. All 58 papers comply with our proposed toolkit definition.

Exemplary Papers. We then identified 10 additional papers published elsewhere, based on exemplary impact (e.g. citations, uptake) such as D3 [14], Piccolo/Jazz [6], and the Context Toolkit [91]. Our total dataset includes 68 papers (Table 1). While other toolkit papers exist, our dataset serves as a representative sample from which we could (1) gather insight and (2) initiate meaningful discussion about evaluation.

Analysis and Results

The dataset was analyzed via several steps. One of the authors conducted open-coding [16] on a subset of our sample, describing the evaluation methods used in each publication. Next, we collectively identified an initial set of evaluation methods and their variations as used across papers. At this point, four other co-authors performed focused coding [16] on the entire sample. We continued to apply the codes to the rest of the sample, iteratively refining and revisiting the coding schema. After coding all papers in our sample, we created categories [16] to derive the overarching evaluation strategies used by toolkit researchers, thus arriving at the four evaluation strategies that we identify as (1) *demonstration*, (2) *usage*, (3) *technical evaluation*, and (4) *heuristic evaluation*. Table 1 summarizes the analysis, showing the count of evaluation strategies seen in our sample. We caution that this frequency count is not necessarily indicative of a strategy's overall appropriateness or success.

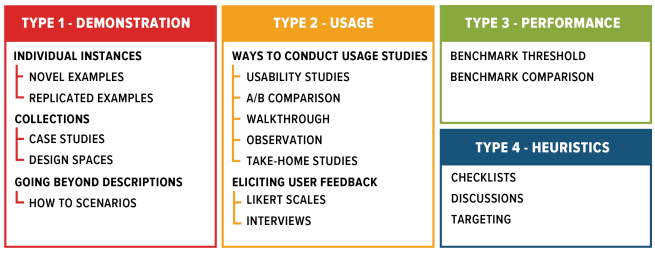

Table 2. A summary of the four evaluation strategies.

The following sections step through the four evaluation types, summarized in Table 2. For each type, we discuss their value and the specific techniques used. We then reflect on challenges for that type, followed by opportunities to strengthen the evaluation: opinions are based on our insights gained from data analysis, our experiences and/or opinions offered by other researchers. The result is a set of techniques that researchers can use, on their own or in combination, to assess claims made about their toolkits.

TYPE 1: DEMONSTRATION

The now famous "*mother of all demos*" by Douglas Engelbart [26] established how demonstrating new technology can be a powerful way of communicating, clarifying and simply showing new ideas and concepts. The transferability of an idea to neighbouring problem spaces is often shown by demonstrating application examples [83]. In our sample, 66 out of 68 papers used demonstrations of what the toolkit can do, either as the only method (19/68) or in combination with other methods (47/68). Demonstrations show *what the toolkit might support*, as well as *how users might work with it*, ranging from showing new concepts [32,91], to focused case studies [4,96] to design space explorations [43,54,64].

Why Use Demonstrations?

The goal of a demonstration is to use examples and scenarios to clarify how the toolkit's capabilities enable the claimed applications. A demonstration is an existence proof showing that it is feasible to use and combine the toolkit's components into examples that exhibit the toolkit's purpose and design principles. These examples can illustrate different aspects of the toolkit, such as using the basic building blocks, demonstrating the workflows, or discussing the included tools. Since toolkits are a 'language' to simplify the creation of new interactive systems [30], demonstrations describe and show how toolkits enable *paths of least resistance* for authoring.

In its most basic form, a demonstration consists of examples exploring the expressiveness of the toolkit by showing a range of different applications. More systematic approaches include explorations of the *threshold*, *ceiling* or *design space* supported by the toolkit. The *threshold* is the user's ability to get started using the toolkit, while *ceiling* refers to how much can be achieved using the toolkit [73]. While demonstrations may not show the full *'height'* of the *ceiling*, they are an indicator of the toolkit's achievable complexity and potential solution space. The principles and goals of the toolkit can also be demonstrated through a design space exploration which enumerates design possibilities [106] and gives examples from different points in that space.

Evaluation Techniques as Used in Demonstrations

Our sample reveals several techniques to demonstrate a toolkit. These techniques are not mutually exclusive and can be combined in different ways. The simplest unit of measurement for demonstration is an *individual instance*. While multiple instances can be described separately, researchers may carefully select instances as *collections* to either explore the toolkit's depth (case studies) or its generative breadth

(design spaces). Toolkit authors may also *go beyond describing* the features of instances, by showing the detailed 'how to' steps involved in the instance authoring process.

Individual Instances

1. Novel Examples. Demonstration of a toolkit can be done by showing the implementation of novel applications, systems or interaction techniques. The Context Toolkit [91] is a classic case of how example applications are used to demonstrate the underlying concepts of *context-awareness* [97]. A more recent example is WorldKit [113], which demonstrates projection-based touch interfaces on everyday surfaces in four different environments. Similarly, in DiamondSpin [98], the authors explore the capabilities of their multi-touch table toolkit by showing five different tabletop designs. Peripheral Displays Toolkit [68] uses three applications to demonstrate ways to enable new peripheral displays. Finally, Sauron [93] describes three prototypes to demonstrate the toolkit's interactive features for physical prototypes. What is important is that these examples detail *how* the features, design principles, and building blocks enable new applications.

2. Replicated Examples. Toolkits often facilitate authoring of systems that were previously considered difficult to build. Recreating prior applications, systems or interaction techniques shows how the toolkit supports and encapsulates prior ideas into a broader solution space. For instance, Prefuse [38] states that they "*reimplemented existing visualizations and crafted novel designs to test the expressiveness, effectiveness, and scalability of the toolkit*". In d.tools [37], the authors recreated a classic iPod interface, while the TouchID Toolkit [67] recreated prior work from external sources (e.g. Rock and Rails [108]) in bimanual interaction. Similarly, SwingStates [2] and Prefab [20] illustrate the expressiveness and power of their toolkit by recreating interaction techniques in the research literature (e.g. Bubble Cursor [34], CrossY [1]). These examples demonstrate how toolkits reduce complexity, effort and development time for recreating applications. Further, replication can demonstrate how the toolkit generalizes across a variety of examples.

Collections

3. Case Studies. Because toolkits often support complex applications, case studies (typically concurrent research projects) can help explore and elaborate the toolkit in greater depth. Five of our 68 papers included case studies to reveal what their toolkit can do. The iStuff toolkit [4] presents case studies of other research projects that use the toolkit. Similarly, the SoD toolkit [96] describes its use in complex case studies: an oil and gas exploration application and an emergency response system. Prefuse [38] reports on the design of *Vizster*, a custom visualization tool for social media data. Although case studies are less common than examples, they convincingly demonstrate the toolkit's application within complex scenarios as opposed to small example applications.

4. Exploration of a Design Space. A design space exploration exemplifies the breadth of applications supported by the toolkit by fitting it into a broader research theme. Design spaces often consist of dimensions with properties (categorical or spectrum variables) [106] that examples can align to. A toolkit author can create a collection of examples that each examine different points in the design space. For example, WatchConnect [43] describes a design space of how the toolkit supports interaction across a watch prototype and a second screen. By providing five examples, including both replicated and novel techniques, the authors satisfy the smartwatch + second screen design space by example. The Proximity Toolkit [64] similarly describes the design dimensions of proxemic interaction [5] (e.g. distance, orientation, identity) and demonstrates through examples how the toolkit enables new proxemic-aware applications. Pineal [54] explores different ways of using and repurposing mobile sensors and outputs to author smart objects, using a combination of novel examples and replication. Finally, DART [61] is an example of a toolkit supporting the exploration of a design space through a range of 'behaviors' and examples. A design space exploration is thus a systematic way of trying to map out possible design boundaries. Although exploring the full design space is often impossible, examples demonstrate the breadth of designs enabled by the toolkit.

Going Beyond Descriptions

5. 'How To' Scenarios. Toolkit papers can demonstrate a step-by-step breakdown of how a user creates a specific application. Scenarios break down tasks into individual steps that demonstrate the workflow, showing the results of each step. We found three ways to describe scenarios. One way is to dedicate a section to describe how one example is authored (e.g. RetroFab [87], Pineal [54]). Second, a scenario can be used throughout the paper to show how different parts of an example come together (e.g. the Proximity Toolkit [64]). Demo scenarios, as in VoodooSketch [12] and Circuitstack [104] are common ways to explain how users might experience a toolkit's *path of least resistance*. Third, authors might include code samples. For instance, Prefuse [38] and Weave [17] use code snippets explaining how certain design principles or building blocks are supported directly in code.

Challenges

Using demonstrations to 'evaluate' a toolkit poses several challenges. First is its rationale: although novel demonstrations built atop the toolkit illustrate toolkit expressiveness, it is sometimes unclear *who* would use such applications and *why*. Second, while creating demonstrations can describe 'what if' scenarios, the demonstration itself may not show that the toolkit can indeed be used by people other than the toolkit's authors. Such lack of external validation may pose issues depending on the claims made in the paper. Third, example applications often aim to implement aspects of a potential future today; however, the target audience might not yet exist or simply be unclear. Speculating on the intended audience creates the risk of an elastic user [18], where the definition of the target audience is stretched to accommodate implementation decisions and toolkit design. Finally, many toolkit systems (e.g. [64, 88, 112]) work with specialized or custom-built hardware. In creating these arrangements, the

authors could alienate the potential audience, as some end-users would not be able to recreate these complicated setups.

Reflection and Opportunities

Provide Rationale for Toolkit Design and Examples. Within every piece of technology lie assumptions, principles and experiences that guide the design of that technology. Many of these assumptions can come across as arbitrary when designing toolkits. However, toolkit authors often rely on their experience even if they do not explicitly mention it. Discussing the understanding of the challenges, perhaps informed by earlier studies or experiences with other tools or toolkits, can help address why different decisions were made. Nebeling et al.'s XD toolkit suite [74,75,76] is a compelling example of how to do this. They constructed several toolkits to structurally and systematically explore the large design space of cross-device computing. They clearly motivated the design and development of each toolkit by earlier experiences in designing toolkits and systems. More generally, research by design [39] helps explore concrete implementations of ideas.

First-Hand Experience. Toolkit authors often have experience creating applications that the toolkit will support, and thus are genuinely familiar with the development challenges and steps that need simplifying. This experience leads to autobiographical design [78] informing the toolkit design process. In Phidgets [32], the authors discuss their frustrations in authoring hardware-based applications, which informed their design and implementation. A toolkit may also leverage experiences with building similar toolkits. The design of D3 [14] evolved from the authors' earlier experiences in creating visualization toolkits (e.g. Prefuse [38], ProtoVis [13]).

Prior Work. Challenges identified in previous research can help motivate the design of toolkits. For instance, the Context Toolkit [91] describes challenges in authoring context-aware applications based on prior work (e.g. new types of sensing from multiple distributed sources).

Formative Studies. Authors can perform formative studies to understand their intended target audience. For instance, in d.tools [37], the authors conducted interviews at product design companies. Understanding current practices can help address challenges with the design of the toolkit.

Discuss Boundaries and Underlying Assumptions. Despite including a 'limitations' section, toolkit authors often do not discuss aspects of the toolkit that do not work well. Critically discussing what does not work or the tasks complicated by the toolkit might help steer away from a 'sales pitch'.

TYPE 2: USAGE

While demonstrations answer the question of '*what can be built with the toolkit*', evaluating *usage* helps verify '*who can use the toolkit*' under certain circumstances, i.e., which tasks or activities can a target user group perform and which ones still remain challenging? To *evaluate* if and how a user group can actually use the tool, it is important to investigate how that user group uses and appropriates the toolkit. Our sample shows that more than half of the papers (35/68) include usage studies. Only one toolkit paper uses a usage study as the only evaluation method [42]. Usage studies are often combined with demonstrations (33/68) or technical evaluations (9/68).

Why Evaluate Usage?

The defining feature of usage evaluations is the involvement of external users working with the toolkit. Much of usage evaluation is informed by traditional user studies [24,53,81], and can help verify whether the toolkit is (1) conceptually clear, (2) easy to use, or (3) valuable to the audience.

Given the prevalence of usability studies in HCI (e.g. [24,81]), many toolkit papers examine the toolkit's *usability* — i.e., how easy it is to use the toolkit. Common measures are users' opinions, preferences, completion time, the number of steps (e.g. lines of code), or number of mistakes. In addition, given that toolkits often propose new workflows, or enable creation of new kinds of artifacts, it is important to know if it will be useful to the target audience. In looking for *utility*, researchers inquire into the audiences' interest or outcomes. One way to assess utility is to look at the output of the toolkit. This consists of investigating the artifacts that the users authored with the toolkit. Lastly, a usage evaluation might look to understand *use* of the toolkit: how a user appropriates a toolkit, how it is used over time, and what kind of workflows are developed. The processes together with the end results can point towards *paths of least resistance,* some which may differ from the ones the toolkit authors' intended.

Evaluation Techniques as Used in Usage Studies

Given the involvement of external people in usage evaluations, toolkit authors can perform a variety of evaluations with users, each yielding different kinds of insights. Our data revealed five *ways to conduct usage studies* and two additional complementary techniques for *eliciting user feedback*. The first four techniques refer to controlled lab experiments, where participants are given consistent tasks that can yield accurate measures, such as completion time. The fifth technique is somewhat more aligned with 'in the wild' studies, which can provide more realism [69,89]. The last two techniques are complementary methods to elicit user feedback.

Ways to Conduct Usage Studies

1. Usability Study. When toolkits claim that they facilitate a process, authors may choose to carry out a usability study. This can help identify issues with the toolkit, using measures of participants' performance (e.g. time, accuracy), and further qualitative feedback. Participants are typically given programming tasks that exploit various aspects of the toolkit. These programming tasks tend to be closed-ended, though some may include a small degree of open-endedness (e.g. [36]). To increase control, some tasks may incorporate prewritten skeleton code (e.g. [74]). Usability studies can examine various aspects of toolkits. For example, Papier-Mâché [52] shows an evaluation of the toolkit's API usability, which revealed inconsistency in the naming of software components and aspects of the toolkit that lacked documentation. Hartmann et al. coined the term "first-use study" [37] in which participants are exposed to a toolkit for the first time and

assigned different tasks. In d.tools [37], the study aimed at determining the *threshold* [73] of the system, while in Exemplar [36] the focus was on determining the successes and shortcomings of the tool. The study in Exemplar [36] combined close-ended tasks with a more open-ended task. Some papers report modifying the toolkit to address issues identified in a usability study [52,60], which Greenberg and Buxton suggest should be the main goal of usability studies [31].

2. A/B Comparisons. One way to suggest improvement over existing work is to compare the new toolkit to a baseline. Baselines include not having a toolkit, or working with a different toolkit. In MAUI [40], the authors compare different platforms to measure what they defined as *effort:* number of classes, total lines of code, lines written for feedthrough and development time. By comparing it to GroupKit (a prior toolkit that supports a similar task [90]) and Java (no toolkit), the authors can show the degree of improvement from the current state-of-the-art. A/B comparisons could test for variations within the toolkit. Lin and Landay [59] compared a full version of their prototyping tool to one without the key features (patterns and layers) to determine the improvement and preference. Finally, both Paperbox [107] and XDStudio [75] compare different configurations of their toolkit.

3. Walkthrough Demonstrations. A walkthrough demonstration consists of showing the toolkit to a potential user and gathering their overall impressions. Unlike *cognitive walkthroughs* [85], walkthrough demonstrations are not about the user working directly with the tool to identify usability problems. In a walkthrough demonstration, the experimenter has full control and explains the workflow to participants, together with examples and even limitations. This approach is particularly suitable when toolkit creators want to get feedback on the utility of their toolkit, as it removes the focus from using the toolkit (as one might find in a usability study) and shifts it towards the value of having the toolkit. While the walkthrough technique has not been explored extensively, RetroFab [87] is an example of this approach. This technique can be useful to gather feedback on the idea rather than the specific toolkit implementation, and might serve for toolkits that are not ready for usability testing or deployment.

4. Observation. Direct observation helps inform how users approached the toolkit to solve problems ranging from closed tasks requiring a specific solution to a given problem, to open tasks where participants formulate the problem and use the toolkit to create their own solution. While our analyzed papers rarely presented any in-depth discussion of participants' processes or workflows, they did provide examples of the toolkit's use. HapticTouch [55] tested participants' ability to transfer concepts about haptics, which were provided at varying levels of abstraction, into an interactive application: its authors assessed the *paths of least resistance* the toolkit afforded to solve both open and close-ended tasks. Our analysis also saw observational studies used within short-term [84] and long-term [51,103] workshop settings involving multiple participants. For example, Pfeiffer et al. [84] asked participants to brainstorm ideas and create Wizard-of-Oz prototypes using the toolkit. Their video analysis discusses the applications created, as well as in-depth details of how their creations were made. In C4 [51], participants attended 3-week workshops, with some staying further for a 4-week artist residency: observation informed its creators on how design decisions held up in the implementation.

5. Take-Home Studies. Some external validity [69] can be acquired by conducting experiments outside lab settings. While it is difficult to deploy a toolkit before it has gained broader acceptance, researchers can provide their toolkit to "early adopter" participants. Participants receive the toolkit (and all necessary components and documentation) to create any applications of their liking within a given timeframe (e.g. a week). Phidgets [32], jQMultiTouch [76] and the Proximity Toolkit [64] are iconic examples where students in an advanced HCI class were given access to the toolkits and necessary hardware components to create interesting examples as a prompt. They all demonstrate how students could easily work with the proposed constructs, where they focused on design aspects of the assignment versus low-level coding.

Eliciting User Feedback

6. Likert Scale Questionnaires. Likert scales provide a non-parametric value pertaining to a question. The questions can later be analyzed either through non-parametric tests or by examining the median values. In toolkit research, while often acting as validation of claims (e.g. ease of use), Likert scales can formalize the results to clarify a hypothesis. For instance, in Exemplar [36], the authors were unsure as to whether the system empowered both experts and non-experts, as the performance between these two can differ considerably. By using Likert scale questionnaires, participant responses confirmed that both experts and non-experts felt empowered, thus validating their hypothesis. Other examples like Damask [59], d.tools [37], Paperbox [107] and Panelrama [114] use Likert scales to quantify user feedback on their system. This feedback often complements other usability results.

7. Open-Ended Interviews. In our sample, 12 papers ask participants about their experiences or challenges performing their tasks, which provided the authors with insight in terms of processes, successes and shortcomings of the toolkit [38,42,114]. Interview questions can start from a script, but are open in that they allow further inquiry as opportunities arise, such as pursuing interesting and/or unclear responses. Quoting participants gives life and adds strength to findings [17,60,95]. Interviews can also expose how users perceive toolkit features, and can contextualize other usage data.

Challenges

Evaluating the toolkit's implementation through usability tests could distract from the conceptual ideas as well as the opportunities facilitated by the toolkit. Olsen [82] warns against falling into "*the usability trap*", as the three underlying assumptions for usability evaluation – walk up and use, standardized tasks, and problem scalability – are rarely met for systems research. Additionally, toolkits in HCI research

are still prototypes. It is difficult for a small team to create a toolkit with the quality of a commercial product (*fatal flaw fallacy* [82]). Controlled experiments measuring usability are limited in scope and evaluate a very small subset of what the toolkit can accomplish, making it difficult to generalize usage results. Furthermore, selected experimental tasks might favour elements that the toolkit can accomplish. In achieving control of the tasks, researchers may optimize for these tasks, or only create what a usability test can measure [82].

While observations of people using the toolkit provide information about use, they may not assess how the toolkit fares in the real world. McGrath [69] discusses this as the trade-off between realism, precision and control. Even in "take home" studies, realism is compromised: participants are given all necessary components, instruction, access to resources (e.g. documentation, direct access to the toolkit creators). This creates an idealistic scenario not necessarily present in real-world adoption [56]. Furthermore, it is difficult to identify appropriate participants for usage evaluations, especially as toolkits propose new ways to solve a problem. Specialized target audiences may not even exist yet [77]. Given the academic context, it is often easiest to find student populations. Students (e.g. computer science students) are often used as a stand-in for the target audience (e.g. developers), assuming that if students can use the toolkit then professionals might too. However, results may not always transfer to the intended target audience. Toolkits often require extensive use before becoming familiar. Thus, a premature evaluation can set up the toolkit for an unfair comparison.

Reflection and Opportunities

Bringing Utility into the Picture. A central challenges of usability evaluation is its focus on toolkit usability *vs.* utility [31]: while a toolkit may be usable, it may not be useful. Walkthroughs and interviews can help here, where questions about utility can be raised and responses explored in depth.

Selecting Tasks and Measures Carefully. While more control, more measures and more quantifiable results seemingly provide rigour, we argue that rigour is only of value if truly representative tasks and appropriate measures are used. Rigour should come from a careful selection of the method, technique, and means of executing the technique. Publications should clearly articulate why the chosen tasks and measures support the claims made in the paper [31].

Recognizing the Consequences of Audience Choice. Toolkit authors should critically reflect and understand the implications of their choice of audience to study. As mentioned, the audience can be a close approximation or a starting point, but authors need to articulate such implications and limitations.

TYPE 3: TECHNICAL PERFORMANCE

While demonstrations and usage studies evaluate *what* a toolkit can do and *who* might use that toolkit, researchers can evaluate the technical performance of the toolkit to find out *how well* it works. From our sample of 68 toolkit papers, about one third of the papers (18/68) include technical performance studies. Technical studies are complementary to demonstration and usage evaluations, as they convey additional information on the technical capabilities of the toolkit.

Why Analyze the Technical Performance?

The goal of studying technical performance is to benchmark, quantify or analyze the toolkit or its components to verify or validate the performance. Technical performance can be measured in terms of efficiency (e.g. speed of the algorithm, throughput of a network protocol), precision (e.g. accuracy of an algorithm, fault tolerance), or comparison against prior techniques. Overall, the purpose is, thus, to measure some form of system performance. These measures show whether it meets basic usage standards (*threshold*), or if there are improvements over the state-of-the-art. Technical benchmarks can push the boundaries of the toolkit to show when it no longer works as expected. Authors sometimes turn to software engineering metrics (e.g. lines of code, number of classes) to show improvement over existing practices.

Techniques as Used in Technical Performance

The Software Engineering community has a rich set of tools to evaluate the performance of systems [9]. Our dataset showed that toolkit authors examine a wide variety of benchmarks (e.g. website loading time [14], spatial resolution [33], framerate [28,51], GPU usage [51], memory allocation [13,51], load time [13], lines of source code [2,91], size of binary [2]). Performance metrics should be tied to the claims of the paper, and the needs that must be satisfied for the toolkit to be operational or go beyond the state-of-the-art.

1. Benchmarking Against Thresholds. For certain types of applications, systems and algorithms, there are known, tested or desirable thresholds that serve as baseline to verify that a system meets a commonly accepted standard of use (e.g. accuracy, latency). For instance, 30 fps is often used for real-time tracking systems [79]. Both KinectArms [28] and EagleSense [112] present new tracking systems benchmarked at this 30 fps rate. Thresholds can be derived empirically, technically or from experience using the tools.

2. Benchmarking Against State-of-the-Art. Benchmarking often looks for improvements over existing state-of-the-art solutions. This comparison approach is often similar to algorithm contributions in HCI (e.g. [110]), where a toolkit's capabilities are compared against well-known baselines, or the best algorithm for that purpose. For instance, in OpenCapSense [33], the authors compared the toolkit's capacitive sensing performance to the earlier CapToolKit [109]. While not a toolkit (and thus not part of our dataset), the $1 Gesture Recognizer [110] is an excellent example of benchmarking against the state-of-the-art: the benchmarks showed that it was considerably close to the state-of-the-art, yet much simpler to implement (about 100 lines of code). D3 [14] compared page load time to a prior toolkit and to Adobe Flash. Page load time was deemed important given their use-case: viewing visualizations created with the toolkit on the web.

Challenges

Technical benchmarks often complement demonstrations or usage studies. Measuring technical benchmarks in isolation may highlight some human aspects of using a toolkit (e.g. frame rate, latency), but do not account for what it is like to use the toolkit. For instance, representative examples may still be difficult to program, even if requiring few lines of code. Similarly, a paper may not always (explicitly) clarify the benchmark's importance (e.g. 30 fps in [112]). Another challenge is that benchmark testing relies on comparisons to an existing baseline. If performance specifications have not already been published, authors must access state-of-the-art systems to perform the comparisons. Given the prototypical nature of HCI toolkits and the *fast-moving targets* of technology [73], many pre-existing baselines may already be deprecated or require extensive reimplementation by the toolkit authors. Alternatively, a baseline may not exist, as the technical challenge may not have been solved before [82].

Reflection and Opportunities

Contextualize and State Technical Limitations. HCI toolkit researchers often have quite different goals from commercial toolkit developers. For example, researchers may want to show how interaction concepts can be packaged within an easy-to-program toolkit (e.g. its API), where the underlying – and perhaps quite limited – infrastructure only serves as proof of concept. Significant limitations should be stated and contextualized to explain why they do not (or do) matter.

Risky Hypothesis Testing. Toolkit authors should openly discuss the rationale behind the tests performed and whether the tests are a form of stress testing. Similar to some of Greenberg and Buxton's arguments [31], perhaps the best approach is to actively attempt to break the toolkit's proposed technical claims (e.g. the ability to accurately track up to four people in real-time [111]) to truly understand the toolkit's technical boundaries. One way to test these boundaries is to stress-test the system's scalability for a chosen metric.

Open Source and Open Access. As toolkit researchers, we can facilitate comparison and replication by making our work available to help future researchers (e.g. [14,64,95]). Ideally, this goes beyond the academic publication or the toolkit source code and documentation, but also includes the benchmarking data so that others can run the tests (e.g. on different computers or as baselines for future studies).

Discuss Implicit Baselines. While a toolkit paper may assume standard metrics to determine that a system works (e.g. 24 fps, or few lines of code to accomplish a task), it may help to mention why this metric is relevant. Thus, less familiar readers can better understand the performance implications.

TYPE 4: HEURISTICS

Heuristics in HCI are typically associated with Nielsen et al.'s (e.g. [72,81]) discount method to informally assess interface usability. Given the challenges of toolkit evaluation, toolkit researchers have devised toolkit-centric heuristics (guidelines) to assess the end-result of a toolkit [10,82]. The toolkit is then inspected against these heuristics, which in turn serves to inform strengths, weaknesses, and reflection of the toolkit's potential value. The heuristics have been extracted from tried and accepted approaches to toolkit design and have been used by others (e.g. Blackwell and Green's heuristics [10] as used by [13,36], Olsen's heuristics [82] as used by [43,58,70,71,74,96]). In our sample, heuristics always complemented other methods.

Why Use Heuristics?

Heuristics are used as a discount method that does not require human participants to gather insight, while still exposing aspects of utility. Olsen's ideas of *expressive leverage* and *expressive match* [82] resonate with Greenberg's view of toolkits as a language that facilitates creation [73], or Myers' themes of successful systems *helping where needed* and creating *paths of least resistance* [73]. Heuristics are based on tried success [72] or theories (e.g. cognitive dimensions [8]).

Blackwell and Green's Cognitive Dimensions of Notation (CDN) [8] was initially offered as a set of discussion points that designers could also use as heuristics to verify system usability. Their primary goal was to create a vocabulary for experts to make early judgements when designing, and to articulate decisions later. The authors describe it as a synthesis of several sources that can partially address elements of the interface design process. CDN also included a questionnaire approach [11] to structure user feedback sessions.

Olsen's heuristics [82] aimed to bring the focus of toolkit evaluation back to what he saw as the value of UI systems research, which corresponds to our aforementioned reasons why HCI researchers build toolkits. Olsen provided terminology and means to support common claims made in toolkit papers. Interestingly, Olsen states that given a set of claims, one can *demonstrate* how the toolkit supports them, which may explain why our data shows prevalent combinations of Type 4 evaluations together with Type 1 (demonstrations).

Following a comprehensive list of heuristics can help identify areas not addressed by the toolkit. Some heuristics might be more crucial (e.g. *problem not previously solved* [82]). Conversely, some may not be relevant for the proposed toolkit (e.g. *secondary notations* [10]). Heuristics can and should be omitted when appropriate [72].

Evaluation Techniques for Heuristics

We identified three ways to carry out a heuristic evaluation: checklists, discussion, and as a basis for usage studies.

1. Checklists. The checklist approach consists of selecting a heuristic evaluation approach and going through individual heuristics one at a time. In doing so, authors can reflect on whether the toolkit satisfies the heuristic or not, and the extent of meeting it. For instance, Hartmann et al. [36] followed Blackwell and Green's CDN through a questionnaire [11]. In evaluating each item, they found that many the limitations of the system were due to the inability to show many sensor visualizations at once. Similarly, Meskens et al. [70] follow

Olsen's heuristics to determine which elements of the interface are lacking (e.g. ability to generalize and reuse).

2. Discussion. In contrast to the checklist approach, Olsen's heuristics [82] are also used as reflection points in the discussion of a toolkit paper. This reflection allows the authors to better understand the limitations and whether there are issues in the toolkit that are not addressed. Both Gummy [71] and WatchConnect [43] are examples of this approach, where authors reflect on shortcomings (and ways to address them) as well as compare their toolkits to the state of the art.

3. Basing Usage Studies on Heuristics. Heuristics can help determine what is useful to evaluate. XDKinect [74] tailored their usage study to some of Olsen's guidelines [82], such as reducing solution viscosity and ease of combination.

Challenges

A danger of heuristic evaluations is falling into self-fulfilling prophecies, where authors stretch definitions of the heuristics to justify their claims. Alternatively, authors might choose to only focus on (1) heuristics that their toolkit addresses or (2) how the toolkit addresses them without acknowledging the negative aspects or compromises (e.g. increasing *flexibility* at the expense of *expressive match*). Sometimes the heuristics are not relevant to a particular toolkit. For example, CDN [10] covers a breadth of applications, where some heuristics only apply to one group (e.g. visual programming environments). Omitting heuristics without clear rationale could lead readers to believe that the authors are cherry picking heuristics. Heuristic evaluations are often carried out by the authors, who may have an implicit bias. While heuristic evaluation in HCI suggests the added value of external evaluators [72,81], it proves difficult for toolkits given their complexity. None of the surveyed papers used external evaluators.

Reflection and Opportunities

Using Heuristics as Design Guidelines. Heuristics can serve complementary purposes: they can inform design as well as help evaluate designs. Thus, toolkit authors can conceptually consider *how* to support aspects of creation early on through best practices (e.g. API practices [99]). As examples, the Intelligibility Toolkit [58] and HapticTouch [55] both discuss heuristics inspiring some of their design goals.

Using Heuristics to Inform Techniques from Prior Types. Given the vocabulary provided by heuristics, authors can consider how demonstrations or usage studies might stem from the heuristics themselves. For example, Olsen [82] suggests that one way to experimentally evaluate *expressive match* is to perform a "design flaw test", where participants are asked to remedy a flaw using a regular design with "good expressive match" (e.g. colour picker) and a deficient design with "bad expressive match" (e.g. hex colour codes).

Transparency. Toolkit authors can disambiguate cherry picking versus ignoring irrelevant heuristics by articulating why a heuristic is or is not considered. This will increase transparency and possibly expose gaps in the evaluation.

DISCUSSION

Our meta-review reveals 4 strategies to *evaluate* toolkits: (1) *demonstrations* (what a toolkit can do), (2) *usage* (who can use the toolkit and how), (3) *technical evaluations* (how well a toolkit performs), and (4) *heuristics* (to what extent the toolkit meets standard guidelines). We now offer several opinions, formed from own toolkit building experiences, the meta-review analysis and other toolkit researchers.

Rethinking Evaluation

Rather than considering some methods as *better* than others, we believe that it is more important to use methods that best match the claims of the toolkit paper, and what that evaluation method might yield. One way to determine this might be for authors to ask themselves: *if the evaluation technique were to be removed, what is the impact to the paper?* In answering that question, authors might realize the essential methods, and which ones are secondary or even unnecessary.

Evaluation by Demonstration?

One central observation in our review is that demonstrations are by far the most common way to communicate the functionality of the toolkit. Demonstrations vary in complexity, ranging from small examples to complex interaction techniques and systems. 19 toolkit papers used demonstration as the only way to communicate or evaluate the toolkit's capabilities. Novel and replicated examples are quite common due to their easy implementation and description. However, further analysis showed that it is rare to find more systematic explorations of the capabilities of toolkits through case studies concurrent to the time of publication, or design space explorations. Moreover, many toolkit papers combine examples with code snippets and how-to scenarios to help the reader understand *what* the toolkit supports. While demonstrations are often not considered a formal *evaluation,* they show evidence through "research by design" [39] and are highly effective in communicating the principles, concepts and underlying ideas of the toolkit. In fact, using the toolkit to create prototypes can lead to refinements in the toolkit itself, as was done in SATIN [41]. When linked back to the five goals of toolkit research, demonstrations provide the most complete and compelling evidence for achieving the goals of designing the new toolkit. The wide adoption of *evaluation by demonstration* indicates that such well explored examples can be a measure of success for the underlying concepts and ideas of a specific toolkit implementation.

Usability Studies (Still) Considered Harmful Some of the Time

Half of all toolkit papers in our sample conducted usage studies. These include compelling examples examining how people work with a toolkit; how a toolkit is used and appropriated in a realistic environment; or how toolkits enable creativity and exploration. Although usage studies play a fundamental role in establishing *who* can use a toolkit, our analysis shows that many authors still fall into the *'usability trap'* [82]. Despite Greenberg and Buxton's warning that usability studies can be '*harmful'* if not applied to the right problem [31], many papers in our sample performed usability studies to evaluate complex toolkits. Such studies may employ artifi-

cial tasks, small sample sizes, and non-representative user groups to evaluate a small subset of paths offered by the toolkit. While still yielding results, these are limited to the specific task, and rarely generalize to the entire toolkit capabilities, development paths, broader audience that would use the toolkit, and the context around toolkit learning and use.

Echoing prior work discussing that usability studies are not always required for toolkit research [45,82], we believe narrow usability studies as currently done by most toolkit authors at best play only a *minor* role establishing or evaluating the novelty or significance of the toolkit and its underlying ideas. If done narrowly, they should at least be combined with other techniques: all but one paper in our sample also included demonstrations or technical evaluations. Even so, we consider this a widespread application of a weak mixed method approach, where researchers may make – perhaps unwarranted – generalized usability claims across the entire toolkit. Careless usability evaluations can be costly, as they may evaluate the wrong possible futures and lead to false conclusions [92]. Usability studies can evaluate parts of the toolkit, but they must be designed and conducted with care.

Successful Evaluation versus Successful Toolkit

In our dataset, we observed a diverse range of toolkits that address various sub-fields within the HCI community, where there is no indication that the success of the toolkit was necessarily tied to the success of the evaluation. Some of these toolkits have had enormous impact within the research community. For example, the Context Toolkit [91] has had a transformative effect on research within the space of context awareness, as evident from the 1326 citations. Other toolkits have moved on to become successful outside of the research community. For instance, D3 [14] has been widely adopted for web-based interactive visualizations. Their paper already suggested that the evaluation may not be indicative of success: "*while we can quantify performance, accessibility is far more difficult to measure. The true test of D3's design will be in user adoption*" [14]. Success can also lie in enabling new research agendas. The Proximity Toolkit [64] operationalized proxemic interaction concepts into concrete building blocks and techniques. Many downloaded the toolkit for research or to learn how to build proxemic-aware applications.

The Need for HCI Infrastructure Research

We started this paper by arguing that toolkits have profoundly influenced HCI research and will continue to do so in the future. Going back to the pioneering work of Engelbart [26], Sutherland [100], or Weiser [105], we observe how invention through building interactive systems, architectures and frameworks enabled them to explore completely new spaces. Since then, there has been an enormous growth in toolkits exploring technical realizations of concepts, techniques and systems in many emerging areas within the field (e.g. physical computing, tangible interfaces, augmented reality, ubicomp) and demonstrating new possible futures.

HCI systems and toolkit research serves to further develop and realize high-level interaction concepts (e.g. proxemic interactions [64]). Consequently, toolkits make these conceptual ideas very concrete, and enable further conversations and follow-up research. For instance, the Context Toolkit [91] was a very successful toolkit that moved research in context-aware computing [97] forward by enabling developers to rapidly prototype context-aware applications. The toolkit provided a component-based architecture separating context inference from the applications that used context information and allowing developers to respond to context changes in an event-driven way. By making these ideas (and their realization in software) very concrete, the Context Toolkit also fueled criticism from researchers who argued that a computational representation of context, as encapsulated in the toolkit, did not capture the complexity of how people behave in the real world. Greenberg [29] argued that many contextual situations are not stable, discernable, or predictable, and argued for context-aware applications to explain the inferred context and how they respond to it (what Bellotti & Edwards refer to as "intelligibility" [7]). Interestingly, these discussions led to development and integration of these ideas in future systems and toolkits, such as the Situations framework [19] and the Intelligibility Toolkit [58].

Limitations

We make no pretense that our overview of evaluation strategies for toolkits is complete. First, to ensure that our meta-review focused on forms of evaluation that are relevant to currently accepted standards, we limited our sample to recently published toolkit papers. Thus, we may have missed forms of evaluation used in past toolkit research. Second, many research projects make multiple contributions not captured in a single paper. Our analysis only reflects what is described in that single paper. For some of the toolkits in our meta-review, additional evaluations were described in later publications (e.g. Prefab [21]). Finally, the authors of this paper have all built and designed toolkits. While our reflection of toolkit evaluation strategies is likely strengthened by our first-hand experience, it may also have introduced bias.

CONCLUSIONS

Research toolkits have fundamentally influenced and shaped the way interactive technology is built, and will continue to do so. Despite the impact and success of toolkits, evaluating them remains a challenge. This paper is a first attempt at clarifying *what* evaluation methods are used, *when* they are appropriate and *how* they are performed. We derived four evaluation types and associated techniques for HCI toolkits based on 68 toolkit papers. We hope our categorization and reflection helps strengthen methods for toolkit research and move technical HCI research forward. Data and other materials can be found at: https://github.com/davidledo/toolkit-evaluation.

ACKNOWLEDGEMENTS

We thank our reviewers and Michael Nebeling for their detailed comments that helped clarify our arguments, and the HCI.Tools workshop participants [66] for ongoing discussions on the value of toolkits for HCI research.

REFERENCES

1. Georg Apitz and François Guimbretière. 2004. CrossY: a crossing-based drawing application. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (UIST '04). ACM, New York, NY, USA, 3-12. http://dx.doi.org/10.1145/1029632.1029635
2. 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, New York, NY, USA, 319-322. https://doi.org/10.1145/1166253.1166302
3. Sriram Karthik Badam and Niklas Elmqvist. 2014. PolyChrome: A Cross-Device Framework for Collaborative Web Visualization. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces* (ITS '14). ACM, New York, NY, USA, 109-118. http://dx.doi.org/10.1145/2669485.2669518
4. Rafael Ballagas, Meredith Ringel, Maureen Stone, and Jan Borchers. 2003. iStuff: a physical user interface toolkit for ubiquitous computing environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '03). ACM, New York, NY, USA, 537-544. http://dx.doi.org/10.1145/642611.642705
5. Till Ballendat, Nicolai Marquardt, and Saul Greenberg. 2010. Proxemic interaction: designing for a proximity and orientation-aware environment. In *ACM International Conference on Interactive Tabletops and Surfaces* (ITS '10). ACM, New York, NY, USA, 121-130. https://doi.org/10.1145/1936652.1936676
6. Ben Bederson, Jesse Grosjean and Jon Meyer. 2004. Toolkit design for interactive structured graphics. *IEEE Transactions on Software Engineering*, 30(8). IEEE 535-546. 10.1109/TSE.2004.44
7. Victoria Bellotti and Keith Edwards. 2001. Intelligibility and accountability: human considerations in context-aware systems. *Hum.-Comput. Interact.* 16, 2 (December 2001), 193-212. http://dx.doi.org/10.1207/S15327051HCI16234_05
8. Michael S. Bernstein, Mark S. Ackerman, Ed H. Chi, and Robert C. Miller. 2011. The trouble with social computing systems research. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '11). ACM, New York, NY, USA, 389-398. https://doi.org/10.1145/1979742.1979618
9. Stephen M. Blackburn, Robin Garner, Chris Hoffmann, Asjad M. Khang, Kathryn S. McKinley, Rotem Bentzur, Amer Diwan, Daniel Feinberg, Daniel Frampton, Samuel Z. Guyer, Martin Hirzel, Antony Hosking, Maria Jump, Han Lee, J. Eliot B. Moss, Aashish Phansalkar, Darko Stefanović, Thomas VanDrunen, Daniel von Dincklage, and Ben Wiedermann. 2006. The DaCapo benchmarks: Java benchmarking development and analysis. In *Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications* (OOPSLA '06). ACM, New York, NY, USA, 169-190. DOI: https://doi.org/10.1145/1167473.1167488
10. Alan Blackwell, Carol Britton, Blackwell, A. Cox, Thomas Green, Corin Gurr, Gada Kadoda, M.S. Kutar et al. 2001. Cognitive dimensions of notations: Design tools for cognitive technology. In *Cognitive technology: instruments of mind* Springer, Berlin, Heidelberg. 325-341. https://doi.org/10.1007/3-540-44617-6_31
11. Alan Blackwell and Thomas Green. 2000. A Cognitive Dimensions questionnaire optimised for users. In *proceedings of the twelfth annual meeting of the psychology of programming interest group.* 137-152.
12. Florian Block, Michael Haller, Hans Gellersen, Carl Gutwin, and Mark Billinghurst. 2008. VoodooSketch: extending interactive surfaces with adaptable interface palettes. In *Proceedings of the 2nd international conference on Tangible and embedded interaction* (TEI '08). ACM, New York, NY, USA, 55-58. http://dx.doi.org/10.1145/1347390.1347404
13. Michael Bostock and Jeffrey Heer. 2009. Protovis: A Graphical Toolkit for Visualization. In *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6. IEEE. 1121-1128. https://doi.org/10.1109/TVCG.2009.174
14. Michael Bostock, Vadim Ogievetsky and Jeffrey Heer. 2011. D^3 Data-Driven Documents. In *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12. IEEE. 2301-2309. https://doi.org/10.1109/TVCG.2011.185
15. Jorge Cardoso and Rui José. 2012. PuReWidgets: a programming toolkit for interactive public display applications. In *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems* (EICS '12). ACM, New York, NY, USA, 51-60. https://doi.org/10.1145/2305484.2305496
16. Kathy Charmaz. 2014. *Constructing grounded theory*. Sage.
17. Pei-Yu (Peggy) Chi and Yang Li. 2015. Weave: Scripting Cross-Device Wearable Interaction. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 3923-3932. https://doi.org/10.1145/2702123.2702451
18. Alan Cooper. 2004. *The inmates are running the asylum: Why high-tech products drive us crazy and how to restore the sanity*. Sams Indianapolis.
19. Anind K. Dey and Alan Newberger. 2009. Support for context-aware intelligibility and control. In *Proceedings*

of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09). ACM, New York, NY, USA, 859-868. https://doi.org/10.1145/1518701.1518832

20. Morgan Dixon and James Fogarty. 2010. Prefab: implementing advanced behaviors using pixel-based reverse engineering of interface structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10). ACM, New York, NY, USA, 1525-1534. https://doi.org/10.1145/1753326.1753554
21. Morgan Dixon, Alexander Nied, and James Fogarty. 2014. Prefab layers and prefab annotations: extensible pixel-based interpretation of graphical interfaces. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (UIST '14). ACM, New York, NY, USA, 221-230. https://doi.org/10.1145/2642918.2647412
22. Pierre Dragicevic and Jean-Daniel Fekete. 2004. Support for input adaptability in the ICON toolkit. In *Proceedings of the 6th international conference on Multimodal interfaces* (ICMI '04). ACM, New York, NY, USA, 212-219. http://dx.doi.org/10.1145/1027933.1027969
23. Brian de Alwis, Carl Gutwin, and Saul Greenberg. 2009. GT/SD: performance and simplicity in a groupware toolkit. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems* (EICS '09). ACM, New York, NY, USA, 265-274. 10.1145/1570433.1570483 http://doi.acm.org/10.1145/1570433.1570483
24. Joseph Dumas and Janice Redish. 1999. *A practical guide to usability testing*. Intellect books.
25. W. Keith Edwards, Mark W. Newman, and Erika Shehan Poole. 2010. The infrastructure problem in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10). ACM, New York, NY, USA, 423-432. https://doi.org/10.1145/1753326.1753390
26. Douglas C Engelbart (1968). The mother of all demos. Presented at ACM/IEEE - *Computer Society's Fall Joint Computer Conference.* San Francisco
27. James Fogarty. Code and Contribution in Interactive Systems Research. In *workshop on HCI.Tools at CHI'2017.*
28. Aaron M. Genest, Carl Gutwin, Anthony Tang, Michael Kalyn, and Zenja Ivkovic. 2013. KinectArms: a toolkit for capturing and displaying arm embodiments in distributed tabletop groupware. In *Proceedings of the 2013 conference on Computer supported cooperative work* (CSCW '13). ACM, New York, NY, USA, 157-166. https://doi.org/10.1145/2441776.2441796
29. Saul Greenberg. 2001. Context as a dynamic construct. *Hum.-Comput. Interact.* 16, 2 (December 2001), 257-268. http://dx.doi.org/10.1207/S15327051HCI16234_09
30. Saul Greenberg. 2007. Toolkits and interface creativity. *Multimedia Tools and Applications*, *32*(2), Springer, 139-159. https://doi.org/10.1007/s11042-006-0062-y
31. Saul Greenberg and Bill Buxton. 2008. Usability evaluation considered harmful (some of the time). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). ACM, New York, NY, USA, 111-120. https://doi.org/10.1145/1357054.1357074
32. Saul Greenberg and Chester Fitchett. 2001. Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology* (UIST '01). ACM, New York, NY, USA, 209-218. http://dx.doi.org/10.1145/502348.502388
33. Tobias Grosse-Puppendahl, Yannick Berghoefer, Andreas Braun, Raphael Wimmer and Arjan Kuijper. 2013. OpenCapSense: A rapid prototyping toolkit for pervasive interaction using capacitive sensing. In *Proc. IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE. San Diego, CA, USA, 152-159. 10.1109/PerCom.2013.6526726
34. Tovi Grossman and Ravin Balakrishnan. 2005. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, New York, NY, USA, 281-290. http://dx.doi.org/10.1145/1054972.1055012
35. Thomas E. Hansen, Juan Pablo Hourcade, Mathieu Virbel, Sharath Patali, and Tiago Serra. 2009. PyMT: a post-WIMP multi-touch user interface toolkit. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (ITS '09). ACM, New York, NY, USA, 17-24. https://doi.org/10.1145/1731903.1731907
36. Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. 2007. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07). ACM, New York, NY, USA, 145-154. https://doi.org/10.1145/1240624.1240646
37. Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. 2006. Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the 19th annual ACM symposium on User interface software and technology* (UIST '06). ACM, New York, NY, USA, 299-308. https://doi.org/10.1145/1166253.1166300

38. Jeffrey Heer, Stuart K. Card, and James A. Landay. 2005. prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, New York, NY, USA, 421-430. http://dx.doi.org/10.1145/1054972.1055031

39. Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. Design science in information systems research. *MIS Q.* 28, 1 (March 2004), 75-105.

40. Jason Hill and Carl Gutwin. 2004. The MAUI toolkit: Groupware widgets for group awareness. *Computing Supported Cooperative Work*, 13, Springer 539-571. https://doi.org/10.1007/s10606-004-5063-7

41. Jason I. Hong and James A. Landay. 2000. SATIN: a toolkit for informal ink-based applications. In *Proceedings of the 13th annual ACM symposium on User interface software and technology* (UIST '00). ACM, New York, NY, USA, 63-72. http://dx.doi.org/10.1145/354401.354412

42. Steven Houben, Connie Golsteijn, Sarah Gallacher, Rose Johnson, Saskia Bakker, Nicolai Marquardt, Licia Capra, and Yvonne Rogers. 2016. Physikit: Data Engagement Through Physical Ambient Visualizations in the Home. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16). ACM, New York, NY, USA, 1608-1619. https://doi.org/10.1145/2858036.2858059

43. Steven Houben and Nicolai Marquardt. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 1247-1256. https://doi.org/10.1145/2702123.2702215

44. Steven Houben, Nicolai Marquardt, Jo Vermeulen, Clemens Klokmose, Johannes Schöning, Harald Reiterer, and Christian Holz. 2017. Opportunities and challenges for cross-device interactions in the wild. *interactions* 24, 5 (August 2017), 58-63. https://doi.org/10.1145/3121348

45. Scott E. Hudson, and Jennifer Mankoff. 2014. Concepts, Values, and Methods for Technical Human–Computer Interaction Research. In *Ways of Knowing in HCI*, Springer New York, NY, USA, 69-93. https://doi.org/10.1007/978-1-4939-0378-8_4

46. Scott E. Hudson, Jennifer Mankoff, and Ian Smith. 2005. Extensible input handling in the subArctic toolkit. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, New York, NY, USA, 381-390. http://dx.doi.org/10.1145/1054972.1055025

47. Stéphane Huot, Cédric Dumas, Pierre Dragicevic, Jean-Daniel Fekete, and Gérard Hégron. 2004. The MaggLite post-WIMP toolkit: draw it, connect it and run it. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (UIST '04). ACM, New York, NY, USA, 257-266. http://dx.doi.org/10.1145/1029632.1029677

48. Martin Kaltenbrunner and Ross Bencina. 2007. reacTIVision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (TEI '07). ACM, New York, NY, USA, 69-74. http://dx.doi.org/10.1145/1226969.1226983

49. Jun Kato, Daisuke Sakamoto, and Takeo Igarashi. 2012. Phybots: a toolkit for making robotic things. In *Proceedings of the Designing Interactive Systems Conference* (DIS '12). ACM, New York, NY, USA, 248-257. https://doi.org/10.1145/2317956.2317996

50. Bonifaz Kaufmann and Leah Buechley. 2010. Amarino: a toolkit for the rapid prototyping of mobile ubiquitous computing. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services* (MobileHCI '10). ACM, New York, NY, USA, 291-298. https://doi.org/10.1145/1851600.1851652

51. Travis Kirton, Sebastien Boring, Dominikus Baur, Lindsay MacDonald, and Sheelagh Carpendale. 2013. C4: a creative-coding API for media, interaction and animation. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction* (TEI '13). ACM, New York, NY, USA, 279-286. http://dx.doi.org/10.1145/2460625.2460672

52. Scott R. Klemmer, Jack Li, James Lin, and James A. Landay. 2004. Papier-Mâché: toolkit support for tangible input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, USA, 399-406. http://dx.doi.org/10.1145/985692.985743

53. Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. 2017. *Research methods in human-computer interaction*. Morgan Kaufmann.

54. David Ledo, Fraser Anderson, Ryan Schmidt, Lora Oehlberg, Saul Greenberg, and Tovi Grossman. 2017. Pineal: Bringing Passive Objects to Life with Embedded Mobile Devices. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, New York, NY, USA, 2583-2593. https://doi.org/10.1145/3025453.3025652

55. David Ledo, Miguel A. Nacenta, Nicolai Marquardt, Sebastian Boring, and Saul Greenberg. 2012. The HapticTouch toolkit: enabling exploration of haptic interactions. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction* (TEI '12), Stephen N. Spencer (Ed.). ACM, New York, NY, USA, 115-122. https://doi.org/10.1145/2148131.2148157

56. David Ledo, Lora Oehlberg and Saul Greenberg. 2017. The Toolkit-Audience Challenge. In *Workshop on HCI.Tools at CHI 2017.*

57. Johnny C. Lee, Daniel Avrahami, Scott E. Hudson, Jodi Forlizzi, Paul H. Dietz, and Darren Leigh. 2004. The calder toolkit: wired and wireless components for rapidly prototyping interactive devices. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques* (DIS '04). ACM, New York, NY, USA, 167-175. http://dx.doi.org/10.1145/1013115.1013139

58. Brian Y. Lim and Anind K. Dey. 2010. Toolkit to support intelligibility in context-aware applications. In *Proceedings of the 12th ACM international conference on Ubiquitous computing* (UbiComp '10). ACM, New York, NY, USA, 13-22. https://doi.org/10.1145/1864349.1864353

59. James Lin and James A. Landay. 2008. Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). ACM, New York, NY, USA, 1313-1322. https://doi.org/10.1145/1357054.1357260

60. James Lin, Mark W. Newman, Jason I. Hong, and James A. Landay. 2000. DENIM: finding a tighter fit between tools and practice for Web site design. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (CHI '00). ACM, New York, NY, USA, 510-517. http://dx.doi.org/10.1145/332040.332486

61. Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2004. DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (UIST '04). ACM, New York, NY, USA, 197-206. http://dx.doi.org/10.1145/1029632.1029669

62. Jennifer Mankoff, Scott E. Hudson, and Gregory D. Abowd. 2000. Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (CHI '00). ACM, New York, NY, USA, 368-375. http://dx.doi.org/10.1145/332040.332459

63. Javier Marco, Eva Cerezo, and Sandra Baldassarri. 2012. ToyVision: a toolkit for prototyping tabletop tangible games. In *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems* (EICS '12). ACM, New York, NY, USA, 71-80. https://doi.org/10.1145/2305484.2305498

64. Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (UIST '11). ACM, New York, NY, USA, 315-326. https://doi.org/10.1145/2047196.2047238

65. Nicolai Marquardt and Saul Greenberg. 2007. Distributed physical interfaces with shared phidgets. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (TEI '07). ACM, New York, NY, USA, 13-20. http://dx.doi.org/10.1145/1226969.1226973

66. Nicolai Marquardt, Steven Houben, Michel Beaudouin-Lafon, and Andrew D. Wilson. 2017. HCITools: Strategies and Best Practices for Designing, Evaluating and Sharing Technical HCI Toolkits. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (CHI EA '17). ACM, New York, NY, USA, 624-627. https://doi.org/10.1145/3027063.3027073

67. Nicolai Marquardt, Johannes Kiemer, David Ledo, Sebastian Boring, and Saul Greenberg. 2011. Designing user-, hand-, and handpart-aware tabletop interactions with the TouchID toolkit. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (ITS '11). ACM, New York, NY, USA, 21-30. https://doi.org/10.1145/2076354.2076358

68. Tara Matthews, Anind K. Dey, Jennifer Mankoff, Scott Carter, and Tye Rattenbury. 2004. A toolkit for managing user attention in peripheral displays. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (UIST '04). ACM, New York, NY, USA, 247-256. http://dx.doi.org/10.1145/1029632.1029676

69. Joseph McGrath. 1995. Methodology matters: Doing research in the behavioral and social sciences. In *Readings in Human-Computer Interaction: Toward the Year 2000 (2nd ed)*. 152-169.

70. Jan Meskens, Kris Luyten, and Karin Coninx. 2010. D-Macs: building multi-device user interfaces by demonstrating, sharing and replaying design actions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology* (UIST '10). ACM, New York, NY, USA, 129-138. https://doi.org/10.1145/1866029.1866051

71. Jan Meskens, Jo Vermeulen, Kris Luyten, and Karin Coninx. 2008. Gummy for multi-platform user interface designs: shape me, multiply me, fix me, use me. In *Proceedings of the working conference on Advanced visual interfaces* (AVI '08). ACM, New York, NY, USA, 233-240. https://doi.org/10.1145/1385569.1385607

72. Rolf Molich and Jakob Nielsen. 1990. Improving a human-computer dialogue. *Commun. ACM* 33, 3 (March 1990), 338-348. 10.1145/77481.77486 http://doi.acm.org/10.1145/77481.77486

73. Brad Myers, Scott E. Hudson, and Randy Pausch. 2000. Past, present, and future of user interface software

tools. *ACM Trans. Comput.-Hum. Interact.* 7, 1 (March 2000), 3-28. http://dx.doi.org/10.1145/344949.344959

74. Michael Nebeling, Elena Teunissen, Maria Husmann, and Moira C. Norrie. 2014. XDKinect: development framework for cross-device interaction using kinect. In *Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems* (EICS '14). ACM, New York, NY, USA, 65-74. http://dx.doi.org/10.1145/2607023.2607024
75. Michael Nebeling, Theano Mintsi, Maria Husmann, and Moira Norrie. 2014. Interactive development of cross-device user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 2793-2802. https://doi.org/10.1145/2556288.2556980
76. Michael Nebeling and Moira Norrie. 2012. jQMultiTouch: lightweight toolkit and development framework for multi-touch/multi-device web interfaces. In *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems* (EICS '12). ACM, New York, NY, USA, 61-70. https://doi.org/10.1145/2305484.2305497
77. Michael Nebeling. Playing the Tricky Game of Toolkits Research. In *workshop on HCI.Tools at CHI'2017.*
78. Carman Neustaedter and Phoebe Sengers. 2012. Autobiographical design in HCI research: designing and learning through use-it-yourself. In *Proceedings of the Designing Interactive Systems Conference* (DIS '12). ACM, New York, NY, USA, 514-523. https://doi.org/10.1145/2317956.2318034
79. Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality* (ISMAR '11). IEEE Computer Society, Washington, DC, USA, 127-136. http://dx.doi.org/10.1109/ISMAR.2011.6092378
80. Mark W. Newman, Shahram Izadi, W. Keith Edwards, Jana Z. Sedivy, and Trevor F. Smith. 2002. User interfaces when and where they are needed: an infrastructure for recombinant computing. In *Proceedings of the 15th annual ACM symposium on User interface software and technology* (UIST '02). ACM, New York, NY, USA, 171-180. DOI: https://doi.org/10.1145/571985.572009
81. Jakob Nielsen. 1994. *Usability engineering*. Elsevier.
82. Dan R. Olsen, Jr.. 2007. Evaluating user interface systems research. In *Proceedings of the 20th annual ACM symposium on User interface software and technology* (UIST '07). ACM, New York, NY, USA, 251-258. https://doi.org/10.1145/1294211.1294256
83. Antti Oulasvirta and Kasper Hornbæk. 2016. HCI Research as Problem-Solving. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16). ACM, New York, NY, USA, 4956-4967. https://doi.org/10.1145/2858036.2858283
84. Max Pfeiffer, Tim Duente, and Michael Rohs. 2016. Let your body move: a prototyping toolkit for wearable force feedback with electrical muscle stimulation. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '16). ACM, New York, NY, USA, 418-427. https://doi.org/10.1145/2935334.2935348
85. Peter G. Polson, Clayton Lewis, John Rieman, and Cathleen Wharton. 1992. Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *Int. J. Man-Mach. Stud.*36, 5 (May 1992), 741-773. http://dx.doi.org/10.1016/0020-7373(92)90039-N
86. Jenny Preece and H. Dieter Rombach. 1994. A taxonomy for combining software engineering and human-computer interaction measurement approaches: towards a common framework. *Int. J. Hum.-Comput. Stud.* 41, 4 (October 1994), 553-583. 10.1006/ijhc.1994.1073 http://dx.doi.org/10.1006/ijhc.1994.1073
87. Raf Ramakers, Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2016. RetroFab: A Design Tool for Retrofitting Physical Interfaces using Actuators, Sensors and 3D Printing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16). ACM, New York, NY, USA, 409-419. https://doi.org/10.1145/2858036.2858485
88. Raf Ramakers, Kashyap Todi, and Kris Luyten. 2015. PaperPulse: An Integrated Approach for Embedding Electronics in Paper Designs. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 2457-2466. https://doi.org/10.1145/2702123.2702487
89. Yvonne Rogers, and Paul Marshall. (2017). Research in the Wild. *Synthesis Lectures on Human-Centered Informatics*, *10*(3).
90. Mark Roseman and Saul Greenberg. 1996. Building real-time groupware with GroupKit, a groupware toolkit. *ACM Trans. Comput.-Hum. Interact.* 3, 1 (March 1996), 66-106. http://dx.doi.org/10.1145/226159.226162
91. Daniel Salber, Anind K. Dey, and Gregory D. Abowd. 1999. The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (CHI '99). ACM, New York, NY, USA, 434-441. http://dx.doi.org/10.1145/302979.303126
92. Antti Salovaara, Antti Oulasvirta, and Giulio Jacucci. 2017. Evaluation of Prototypes and the Problem of Possible Futures. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI

‘17). ACM, New York, NY, USA, 2064-2077. https://doi.org/10.1145/3025453.3025658

93. Valkyrie Savage, Colin Chang, and Björn Hartmann. 2013. Sauron: embedded single-camera sensing of printed physical user interfaces. In *Proceedings of the 26th annual ACM symposium on User interface software and technology* (UIST ‘13). ACM, New York, NY, USA, 447-456. http://dx.doi.org/10.1145/2501988.2501992
94. Valkyrie Savage, Sean Follmer, Jingyi Li, and Björn Hartmann. 2015. Makers’ Marks: Physical Markup for Designing and Fabricating Functional Objects. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (UIST ‘15). ACM, New York, NY, USA, 103-108. https://doi.org/10.1145/2807442.2807508
95. Valkyrie Savage, Xiaohan Zhang, and Björn Hartmann. 2012. Midas: fabricating custom capacitive touch sensors to prototype interactive objects. In *Proceedings of the 25th annual ACM symposium on User interface software and technology* (UIST ‘12). ACM, New York, NY, USA, 579-588. https://doi.org/10.1145/2380116.2380189
96. Teddy Seyed, Alaa Azazi, Edwin Chan, Yuxi Wang, and Frank Maurer. 2015. SoD-Toolkit: A Toolkit for Interactively Prototyping and Developing Multi-Sensor, Multi-Device Environments. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces* (ITS ‘15). ACM, New York, NY, USA, 171-180. https://doi.org/10.1145/2817721.2817750
97. B. Schilit, N. Adams, and R. Want. 1994. Context-Aware Computing Applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications* (WMCSA ‘94). IEEE Computer Society, Washington, DC, USA, 85-90. http://dx.doi.org/10.1109/WMCSA.1994.16
98. Chia Shen, Frédéric D. Vernier, Clifton Forlines, and Meredith Ringel. 2004. DiamondSpin: an extensible toolkit for around-the-table interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI ‘04). ACM, New York, NY, USA, 167-174. http://dx.doi.org/10.1145/985692.985714
99. Jeffrey Stylos and Brad A. Myers. 2008. The implications of method placement on API learnability. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering* (SIGSOFT ‘08/FSE-16). ACM, New York, NY, USA, 105-112. http://dx.doi.org/10.1145/1453101.1453117
100. Ivan Edward Sutherland. 1980. *Sketchpad: A Man-Machine Graphical Communication System.* Garland Publishing, Inc., New York, NY, USA.
101. Nicolas Villar, Kiel Mark Gilleade, Devina Ramdunyellis and Hans Gellersen. 2007. The VoodooIO gaming kit: a real-time adaptable gaming controller. *Comput. Entertain.* 5, 3, pages. http://dx.doi.org/10.1145/1316511.1316518
102. Nicolas Villar, James Scott, Steve Hodges, Kerry Hammil, and Colin Miller. (2012) .NET Gadgeteer: A Platform for Custom Devices. *In Pervasive Computing. Pervasive 2012. Lecture Notes in Computer Science*, vol 7319. Springer, Berlin, Heidelberg. 216-233 https://doi.org/10.1007/978-3-642-31205-2_14
103. Akira Wakita and Yuki Anezaki. 2010. Intuino: an authoring tool for supporting the prototyping of organic interfaces. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems* (DIS ‘10). ACM, New York, NY, USA, 179-188. http://dx.doi.org/10.1145/1858171.1858204
104. Chiuan Wang, Hsuan-Ming Yeh, Bryan Wang, Te-Yen Wu, Hsin-Ruey Tsai, Rong-Hao Liang, Yi-Ping Hung, and Mike Y. Chen. 2016. CircuitStack: Supporting Rapid Prototyping and Evolution of Electronic Circuits. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (UIST ‘16). ACM, New York, NY, USA, 687-695. https://doi.org/10.1145/2984511.2984527
105. Mark Weiser, (1991). The Computer for the 21st Century. *Scientific American*, *265*(3), 94-105.
106. Mikael Wiberg and Erik Stolterman. 2014. What makes a prototype novel?: a knowledge contribution concern for interaction design research. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational* (NordiCHI ‘14). ACM, New York, NY, USA, 531-540. https://doi.org/10.1145/2639189.2639487
107. Alexander Wiethoff, Hanna Schneider, Julia Küfner, Michael Rohs, Andreas Butz, and Saul Greenberg. 2013. Paperbox: a toolkit for exploring tangible interaction on interactive surfaces. In *Proceedings of the 9th ACM Conference on Creativity & Cognition* (C&C ‘13), Ellen Yi-Luen Do, Steven Dow, Jack Ox, Steve Smith, Kazushi Nishimoto, and Chek Tien Tan (Eds.). ACM, New York, NY, USA, 64-73. 10.1145/2466627.2466635 http://doi.acm.org/10.1145/2466627.2466635
108. Daniel Wigdor, Hrvoje Benko, John Pella, Jarrod Lombardo, and Sarah Williams. 2011. Rock & rails: extending multi-touch interactions with shape gestures to enable precise spatial manipulations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*(CHI ‘11). ACM, New York, NY, USA, 1581-1590. https://doi.org/10.1145/1978942.1979173
109. Raphael Wimmer, Matthias Kranz, Sebastian Boring and Albrecht Schmidt. 2007. A Capacitive Sensing Toolkit for Pervasive Activity Detection and Recognition. In *Proc. International Conference on Pervasive Computing and Communications (PerCom’07)*, IEEE, White Plains, NY., 171-180. 10.1109/PERCOM.2007.1

110. Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology* (UIST '07). ACM, New York, NY, USA, 159-168. https://doi.org/10.1145/1294211.1294238

111. Jacob O. Wobbrock and Julie A. Kientz. 2016. Research contributions in human-computer interaction. *interactions* 23, 3 (April 2016), 38-44. https://doi.org/10.1145/2907069

112. Chi-Jui Wu, Steven Houben, and Nicolai Marquardt. 2017. EagleSense: Tracking People and Devices in Interactive Spaces using Real-Time Top-View Depth-Sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, New York, NY, USA, 3929-3942. https://doi.org/10.1145/3025453.3025562

113. Robert Xiao, Chris Harrison, and Scott E. Hudson. 2013. WorldKit: rapid and easy creation of ad-hoc interactive applications on everyday surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13). ACM, New York, NY, USA, 879-888. https://doi.org/10.1145/2470654.2466113

114. Jishuo Yang and Daniel Wigdor. 2014. Panelrama: enabling easy specification of cross-device web applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 2783-2792. http://dx.doi.org/10.1145/2556288.2557199

115. Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. 2009. Sikuli: using GUI screenshots for search and automation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (UIST '09). ACM, New York, NY, USA, 183-192.