Concepts, Values, and Methods for Technical Human–Computer Interaction Research

Scott E. Hudson and Jennifer Mankoff

This chapter seeks to illuminate the core values driving technical research in human–computer interaction (HCI) and use these as a guide to understanding how it is typically carried out and why these approaches are appropriate to the work. HCI overall seeks to both understand and improve how humans interact with technology. Technical HCI focuses on the technology and improvement aspects of this task—it seeks to use technology to solve human problems and improve the world. To accomplish this, the fundamental activity of technical HCI is one of *invention*—we seek to use technology to *expand what can be done* or to *find how best to do things* that can already be done. Inventing new solutions to human problems, increasing the potential capabilities of advanced technologies, and (in a spiral fashion) enabling others to invent new solutions and/or apply advanced technical capabilities are all central to technical HCI. The ability to create new things, to mold technology (and the world), and to enhance what people (or technology) can do drives our fascination with technical work; hence, the core value at the heart of technical HCI is invention.

One way of understanding the work of technical HCI research is by contrasting it with other types of HCI research. In an interdisciplinary setting such as HCI, we often shift between disciplines that have stable and functional but potentially contradictory world views. In doing so, we are confronted with the need to select and use (or at least appreciate, understand, and evaluate) a wide range of methods and with them a wide range of expectations and values. For example, different disciplines, such as social and cognitive psychology, design, and computer science, have evolved their own methods, value systems, and expectations about what constitutes appropriate and impactful work. Because they work well for individual disciplines, these expectations and values are often left somewhat unexamined within the discipline itself. For a researcher to work effectively within a discipline, it is critical to know and heed these expectations and values (and hence be able to distinguish and

S.E. Hudson (✉) • J. Mankoff
Human–Computer Interaction Institute, School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA
e-mail: scott.hudson@cs.cmu.edu; jmankoff@cs.cmu.edu

J.S. Olson and W.A. Kellogg (eds.), *Ways of Knowing in HCI*,
DOI 10.1007/978-1-4939-0378-8_4,

produce good vs. not-so-good work). But in turn it is often less critical to fully understand the variety of perspectives held in other disciplines. However, in an interdisciplinary setting like HCI, examining *why* particular methods are suited to particular kinds of work is important. While invention is not unique to technical HCI (this is also a clear component of design-oriented HCI; see Chaps. Science and Design and Research Through Design in HCI), this distinction does separate it from parts of HCI that aim to describe or understand the world through, for example, new discoveries about how the world works, critical theory, or models of human behavior. Thus, as we lay out the expectations, values, and approaches inherent in technical HCI, we will use as a touchstone the contrast between its main activity of *invention* with the focus on *discovery* that typifies some approaches to HCI research.

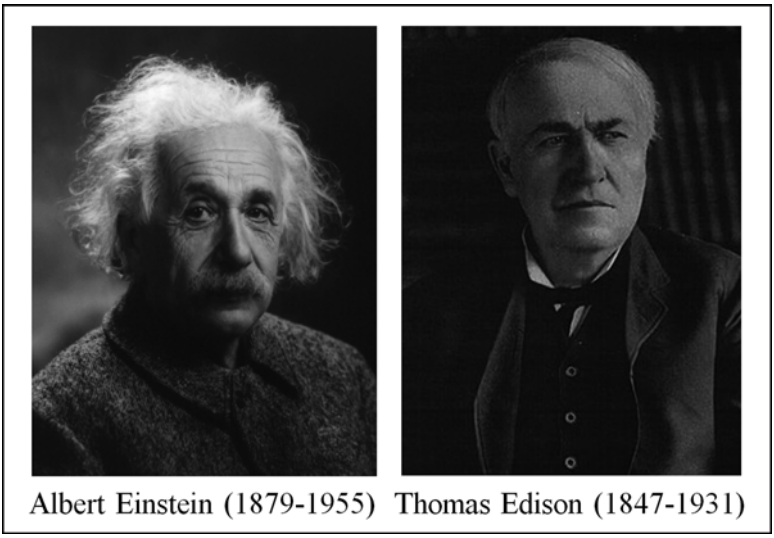
Another way of understanding technical HCI research is by contrasting it with other types of technical work that is not research. For our purposes, *research* can be seen as having *the creation of reusable knowledge* at its core. More specifically *technical HCI research* emphasizes knowledge about how to create something (invention) but also knowledge that might be reused to assist in the creation of a whole class of similar things or even multiple types of different things. For example, several decades ago considerable research effort went into developing ways to allow graphical user interface (GUI) layout to be specified graphically, including the first modern "interface builder" (Hullot, 1986). In contrast, *development* has at its core the creation of a particular thing—something we might often consider a *product*. Development generally requires creation of knowledge, but there is no particular need that this knowledge be reusable. So for example, numerous similar "interface builder" tools are now available in various development environments. Each of these required substantial effort to create and perfect. But only a small part of those efforts have produced reusable concepts.

The distinction between research and development leads to differences in approach among those who practice purer versions of each. However, there is no clear dividing line between them. For example, development of nearly any useful artifact can provide knowledge about how to (or perhaps how not to) build another similar artifact. Further, as will be considered later in this chapter, good evaluation of inventive research almost always mandates some development work—the building of some, or all, of the thing invented. In the end, this means that research and development activities are often intertwined and can be difficult to cleanly separate. Thus, in the second half of this chapter we describe the work of invention in HCI, focusing on types of impact, the essential role of development in validating any invention (through a proof-of-concept implementation), and other forms of validation.

Einstein Versus Edison: Invention as the Basis of Technical HCI Work

Activities of *invention* at their core seek to bring useful new things into the world. This nearly always requires knowing facts about the world and may entail pursuit of new discoveries if the necessary facts are not known or not known well enough. But the heart of invention is *changing how the world works through innovation and creation*.

This is the core purpose and the typical passion of those who undertake activities of invention. In contrast, activities of *discovery* at their core seek to *develop new understandings of the world*. To the extent that inventions play a role in these activities, they are in the service of discovery.



Photos are in the public domain

Note that we might have used the terms *science* and *engineering* here, rather than discovery and invention. In our view, discovery and invention are at once more descriptive and more neutral terms. Both activities are critically important to the success of HCI, but there is a discernible bias, at least in academic circles, towards science and away from engineering. We can see this by noting that we often hear phrases such as "Where's the science in this?" and "That's just engineering," but we pretty much never hear "Where's the engineering in this?" or "That's just science." In fact "science" is often misused as a synonym for "rigorous" or just "good work" irrespective of whether the work is actually scientific in nature. On the other hand, both discovery and invention can confer great benefits to society, and as such both have been honored. We can see this by noting that exemplars such as Einstein and Edison are both held in high regard in many societies.

There are many similarities in the work of discovery and invention but also some key differences. These have to do with the underlying values and goals of each type of work, specifically what constitutes how work in the field moves forward and what constitutes a trustworthy and valuable result.

Differences in How Fields Move Forward

Activities of discovery can have a variety of aims, including generating rich, empirically based descriptions, and creating new theoretical understandings

(see Chaps. Reading and Interpreting Ethnography and Grounded Theory Method, this volume). Once articulated, theories typically form *framing truths* that establish a context for the work. The work of discovery often proceeds by elaborating and refining these framing truths to progress towards improved understandings. An initial theory that explains the most easily observable facts may be refined to explain more phenomena or to be more predictive. This progression requires developing and testing competing ideas (which might both be consistent with a framing theory). For example, both Newtonian and Einsteinian notions of gravity explain everyday objects falling to earth, and even the motion of planets, quite well. Only when we consider finer and more difficult-to-observe phenomena does one clearly improve on the other. As another example, the speed and accuracy of directed reaching movements are well described in one dimension by Fitts' law (Fitts, 1954). However, this theory has various limits (for example, when applied to 2D targets of arbitrary shape). Newer theories, for example those based on the microstructure of movements (see Meyer, 1990), provide a more detailed account of the same phenomena and allow us to overcome some of these limitations (see for example, Grossman & Balakrishnan, 2005b; Grossman, Kong, & Balakrishnan, 2007).

In contrast, activities of invention almost always progress towards the creation of new or better things but not necessarily through refinement. Normally we invent by combining a set of things we already understand how to create into larger, more complex, or more capable things that did not previously exist. The early phonograph for example made use of existing concepts such as a mechanism for rotary motion at a carefully controlled rate and the use of a horn shape for directing sound and combined these with a new method for recording and reproducing small vibrations with a needle in a trace scored in a tinfoil sheet. Similarly, in an HCI context the first graphical interfaces (Sutherland, 1963) were created using existing input and display devices (a light pen, buttons, rotary input knobs, and a random dot CRT) along with new concepts expressed in software to create (among other pioneering advances) the ability of users to manipulate objects displayed graphically by pointing at them. In both inventions each of the detailed precursors was combined to create a much more complex and functional whole based on these smaller and simpler parts. In some cases, activities of invention may start with a larger truth (about something that *should* be possible), but the detailed process of invention still typically depends on the combining of smaller or simpler parts into a larger and more complex whole. Hence, in contrast to discovery, as we progress in invention we are not necessarily refining a framing truth. In fact, our understanding can sometimes actually decrease because we are creating more complex things that are less well understood than the simpler things they are made of. However, the things created are more *capable*—they do more or better things in the world—and this is the core of inventive progress.

Differences in What Makes a Result Valuable and Trustworthy

In discovery work, the properties of valuable and trustworthy results are intertwined. Core values in discovery work include increasing understanding (e.g., of new phenomena) or

understanding in more powerful ways (e.g., more profoundly or in some cases predictively). But the desire to *know* and have confidence in results makes the details and reliability of the methods used to reach a result of central importance (what Gaver calls "epistemological accountability" in Chap. Science and Design). In some sense, the methods used to obtain a result are part of the result. The assertion of an understanding about the world cannot stand on its own; it is necessary to know about the method (or in some perspectives, the person; see Chap. Reading and Interpreting Ethnography).

The need for high confidence in results drives the familiar tactic of isolating and testing a small number of variables—often just one or two—in an attempt to separate their effects from other confounds. This tactic achieves increased trustworthiness at the cost of focusing on less complex circumstances. As a result, a study that tests a theory in a specific context may only be able to make claims about a narrow slice of reality. This can make it hard to generalize to more complex, real-world settings without replicating the study in many different but similar settings to be sure that the underlying theory is robust across changing circumstances. To be sure, some forms of discovery grapple more directly with complexities of the real world (see many chapters herein), but confidence in the results, building consensus, and causal attribution can be more difficult.

Invention, in contrast, privileges the value of creating something that has the potential for practical impact. To improve practicality, inventions are most valued if they work within the full complexity of the world around us. In fact, in many cases, if we limit the scope of work to very controlled situations (e.g., with only one or two degrees of freedom), it can easily *destroy* the value of the work. Often we start with specifics and use them to create something that has multiple uses. Indeed to the extent it is possible to apply the result (the invention) in multiple domains it may be considered more valuable.

For invention, the goodness of a result is a property of the concept invented. The properties of the thing invented generally stand alone and can be understood and evaluated independently of the particular methods used in the inventive process. It might be that the inventors came up with their result by means of an arduous process of testing many alternatives, or it might be that the concept came to them in a dream the night before. However, if both paths lead to the same invention, it is equally good. The trustworthiness of an inventive result depends on an examination of the thing that was invented (almost always through consideration of an implementation of it).

The Work of Invention in Technical HCI

We have shown that invention can be seen as an activity that creates artifacts that can solve problems in the world and that the things that make a result trustworthy and valuable differ between activities of invention and discovery. In this section we explore the process of invention, focusing on key aspects of technical HCI research.

Our focus in this section is not on the creative process per say, but rather on the directions from which one might approach invention.

We begin by reviewing the types of contributions typically found in technical HCI research (direct creation and enabling research). Next we review approaches to concept creation, followed by proof-of-concept implementations, the core form of validation for invention. This form of validation is a crucial and inseparable part of the process of concept creation. However, while building takes on a central role, additional validations may help to show the specific impacts of different types of technical contributions. We then present a review of these types of secondary validations. Thus we might well break up the work of inventive research into three parts rather than two: concept creation, proof-of-concept implementation, and (additional) validation.

Types of Contributions

The contributions that can be made by inventive HCI research can come in a number of forms. Many of them might be summed up at the highest level as supporting the invention of things that meet human needs. This can in turn be separated into at least two overall categories: *direct creation* of things meeting human needs and development of things that *enable* further invention.

Direct creation is most straightforward. This might involve creation of something that improves some aspect of a long-standing goal such as supporting collaborative work at a distance (Engelbart & English, 1968; Ishii, Kobayashi, & Arita, 1994) or selecting items on a screen more quickly (Sutherland, 1963; Grossman & Balakrishnan, 2005a); that introduces a new capability such as interacting with wall displays that are larger than the reach of a person's arms (Khan et al., 2004; Shoemaker, Tang, & Booth, 2007); or that brings a capability to a new user population such as photography by the blind (Jayant, Ji, White, & Bigham, 2011).

Enabling research on the other hand is more indirect. It has as a goal not directly addressing an end-user need, but rather to enable others to address a need by making it possible, easier, or less expensive for future inventive work to do so. Enabling research can also come in a number of forms. These include development of tools, systems, and basic capabilities.

Tools generally seek to make it much easier to create a certain class of things. Tools normally do not directly meet end-user needs. Instead, they act indirectly by enabling developers to quickly and easily meet end-user needs or to construct complex and functional artifacts. For example, through extensive UI tools research in the 1980s (such as Buxton, Lamb, Sherman, & Smith, 1983, Cardelli, 1988), specifying the appearance and basic functioning of a GUI is now a simple enough matter that it can often be done by those with only minimal programming ability. Tools also often bring a benefit of making it practical to create a broader set of things. For example, subArctic (Hudson, Mankoff, & Smith, 2005) and Amulet (Myers et al., 1997) are GUI toolkits that provide high-level abstractions that make it much easier

to build interactive systems. Tools may not provide any new capabilities at all, but instead make existing capabilities much more accessible or useful for developers (see threshold and ceiling effects, below).

Systems bring together a set of capabilities into a single working whole—often providing abstractions that make these capabilities more useful, more manageable, and/or easier to deploy or reuse. For example, the input handling abstractions in the Garnet toolkit (Myer, 1990) made use of finite state machines for controlling interaction as many systems do—something already widely used. However, it provided a new highly parameterized abstraction of that concept which made it much easier for developers to use. Systems also sometimes bring together a disparate set of capabilities that has not been combined before or combine capabilities in new ways that make them more useful. As an example, every major operating system today includes a subsystem specifically for handling overlapping windows, which provides basic input and output capability on a single set of devices that can be shared by many programs.

Basic capabilities: Another enabling contribution is an advance on a specific and difficult problem that is holding up progress in a problem domain. The advance made may be very narrow but have value in the breadth of the things it enables. By creating new or improved algorithms, new circuits, or new sensors, we can enable a range of new inventions. Examples of HCI-relevant basic capacities that have been introduced, e.g., to modern operating systems include input device drivers, event modeling (providing an abstraction that describes user input in a device-independent fashion), and graphics systems (which provide an abstraction for displaying images on a screen; typically one that can be transparently translated into a range of fast graphics hardware). In another example, algorithms for face recognition and tracking that were able to operate at frame rate (Viola & Jones, 2001) enabled a range of new capabilities such as digital cameras that automatically focus on faces, thus producing better photography by average consumers with no additional effort on their part.

Finally, it is important to note that enabling research also often takes the form of *importing* and *adapting* advances made in other technical areas and putting them to use for new purposes. In some respects this might not be considered invention per se. However, it surely must be considered a research advance, as in the modern world substantial progress is made in exactly this fashion—an idea or a concept originally created in one research domain is first imported, and then typically adapted, for use in others. For example, finite-state automata are now heavily used in implementing interaction techniques. This concept was first introduced for HCI use by Newman (1968). However, Newman clearly did not invent finite-state automata (they were originally devised to model neuronal activity (McCulloch & Pitts, 1943) and subsequently used in many other ways). Nonetheless, the idea has been of great benefit in user interface implementation and has since been built on and improved upon numerous times (Wasserman, 1985; Jacob, 1986; Appert & Beaudouin-Lafon, 2008; Schwarz, Mankoff, & Hudson, 2011). As such this importing and adaptation of a powerful technique can have great value and so must be considered a contribution in its own right.

Approaches to Concept Creation

It is extremely difficult to put one's finger on the best approach to inventing a new concept. However, there are some strategies that have been shown to be productive in our experience. One of the most frequent outcomes of inventive work in HCI is to devise a new way to bridge between technical capabilities and human needs. This simple framing points the way to some of the most common strategies for developing technical contributions. A researcher can start from an observed human need and seek to find a technical approach that can make a positive impact on the need. This approach often leads one to specialize in one or more application areas, learning more and more about the details of human needs in that area. For example, systems supporting special-needs populations such as elder care (see for example Mynatt, Essa, & Rogers, 2000; Mynatt, Rowan, Craighill, & Jacobs, 2001) have often taken this approach. A researcher may do discovery-based work to better understand these needs (and human properties that impact them) and then seek (mostly existing) technological capabilities that might be used to meet these needs.

Within this general framework, one can also work from the technology side: a researcher may specialize in one or more areas of useful or promising technology—learning a substantial amount about how they work (and/or where their weaknesses lie), and extending and improving them, and then seeking to find existing human needs that the technology might have a positive impact on. For example Shwetak Patel and his colleagues have produced several related types of sensors that work by observing changes in the noise found on household power lines (see Patel, Robertson, Kientz, Reynolds, & Abowd, 2007; Cohn, Morris, Patel, & Tan, 2011; Gupta, Chen, Reynolds, & Patel, 2011). This work was undertaken not because of a human need but because of a new technological opportunity that the researchers have considerable expertise with (the ability to rapidly analyze and classify minute variations in "noise" as an intentional signal). Initially, the research was used to sense the location of people within the home, but the researchers also developed the capability to sense appliance use and then simple gestures. These potentially very useful sensing capabilities could be installed simply by plugging a device in (as opposed to hiring an electrician). Thus, as it happened, the resulting product was able to meet several human needs, once it was packaged in an easily deployable box and tied to applications of interest.

This type of technology-first approach has developed a bad reputation within the HCI research community. Historically, researchers coming from technological disciplines have not always matched their emphasis on progress in the technology with careful attention to true human needs. However, if inventions are in the end really valued in proportion to their positive effect on human needs, then it does not fundamentally matter whether a technology-driven or a needs-driven approach was driving the effort to meet those needs. Not only that, technology is currently changing very quickly, while human needs are changing relatively slowly. Indeed, technology is becoming pervasive so rapidly that it is beginning to drive change in human needs. Also, invention that focuses ahead of the technology curve is more likely to

be relevant in the 5–10-year horizon that matters in research. These factors combine to make technology-first invention an effective way to build bridges between technology and human needs.

Of course, in practice good researchers often do not limit themselves to either pure needs-first or technology-first approaches. Instead a common approach is to study (or simply stay informed about) the properties of people and the progress in meeting needs within a few application areas and at the same time carefully track progress in a range of potentially useful technologies, searching for new things that might meet outstanding needs. This points to another important property of inventive work—that progress is very often made not by conceiving of entirely new things but instead by *recognizing* that innovations might be used in additional ways and adapting or combining them to meet existing needs. While we often think of invention at its heart as the conception of new things, in fact it much more often involves recognition of new possibilities within already invented things or enabled by new combinations of things (followed in many cases by some adaptation). For example, low-cost MEMS-based accelerometers were originally marketed in large part to support the deployment of airbags in automobiles. But once these devices became available, they were adapted for HCI use. First they were used for exploring the use of tilt as a general form of input (Harrison, Fishkin, Gujar, Mochon, & Want, 1998). This in turn was adapted in additional research on the use of sensors in mobile devices to support landscape/portrait display orientation switching (Hinckley, Pierce, Sinclair, & Horvitz, 2000), which was in turn adopted with small modifications in most current smartphone and tablet interfaces.

In addition to bridging between technology and needs, another typical strategy for making progress is to seek out particular roadblocks to advancement and focus specifically on those. This strategy typically involves carefully tracking progress in some application or technological area, analyzing what the roadblocks to progress or limitations of current solutions are, and then producing concepts targeted specifically at these roadblocks. This approach can often be more indirect—it does not seek to directly impact a human need but instead enables something else that (eventually) will. For example, the authors' joint work on tools and techniques for dealing with uncertainty (Mankoff, Hudson, & Abowd, 2000a, 2000b; Schwarz, Hudson, & Mankoff, 2010b) arose in part from the difficulty of building a specific recognition-based interface to address the need of people with certain disabilities to use something other than the keyboard and mouse for computer input. Tools are a common outcome of this paradigm.

Validation Through Building of Proof-of-Concept Implementations

When we consider validation of an invented concept there are many criteria with which we might judge it. However, most fundamental is the question of "does it work?". A concept can have many good properties, but unless and until it can be

realized in a form that actually functions, none of those properties matter very much. Further, experience with invented concepts shows that many ideas that seem excellent at the early point we might call *on paper* fail in the details that they must confront during implementation. That is, there are one or more seemingly small or hidden details that end up becoming a major obstacle to practical implementation of the concept. Most small details are relatively unimportant. However, some details can end up critically important, and experience has clearly shown that it is very difficult to segregate the critical from the trivial details in advance. This difficulty leads to the most fundamental of validation approaches for inventive work: *proof-of-concept implementation*. Because of the difficulty of uncovering critical details, experienced inventors do not put much credence in an idea until it has been at least partly implemented; in short: *you do not believe it until it has been built*.

The centrality of proof-of-concept implementations as a validation mechanism is so strong that the evolved value system gives *building* a central role. Even a really strong user study or other empirical evaluation cannot improve a mediocre concept (or tell us how good an invention it is). In contrast, a proof-of-concept implementation is a critical form of validation because an invented concept is not normally trusted to be more than mediocre without an implementation.

While the creation of concepts is arguably the most important aspect of invention, proof-of-concept implementations typically consume the most time and effort in inventive work. Building things is typically hard, so hard that it is often impractical to build a complete implementation of a candidate concept. This should not be surprising since it is not uncommon to spend millions of dollars and years of time on the development of a significant real-world product. However, it makes little sense to expend the resources necessary to create a complete implementation of a concept before much is known about how well, or even whether, it might work. Hence, in research most proof-of-concept implementations are compromises that implement some of the critical aspects of an idea but do not necessarily consider all the different factors that must be addressed for a full complete product. Such a compromise seeks to maximize the knowledge gained while working within appropriate constraints on the resources required for building.

Questions Proof-of-Concept Implementations Answer

Proof-of-concept implementations normally seek to elicit particular types of knowledge. This knowledge most often starts with some variation on the basic question of "does it work?". However, we often end up asking "does it work well enough?". How we choose to define "well enough" in turn has a strong impact on the type and extent of implementation we undertake. Sometimes we are looking for evidence indicating that the concept offers some advantage over existing solutions to the same problem. For example there were a number of promising input devices for pointing at displays devised before the mouse (English, Engelbart, & Berman, 1967), but the mouse was found to be a particularly good pointing device compared to its competitors (Card, English, & Burr, 1978). Sometimes, particularly when

creating a completely new capability or overcoming a critical stumbling block, we are only looking for evidence that the concept works at a minimal level (but perhaps shows promise to be improved). An example is our exploration of the value of cords as an input device (Schwarz, Harrison, Hudson, & Mankoff, 2010a). Sometimes we require information about accuracy, accessibility, or effectiveness of the technical concepts with respect to end users of some type, in which case a certain level of robustness may be required.

The question of "does it work (well enough)?" is also complicated by the fact that the inventions most valued are often those that are most robust to the widely varying conditions of the real world. Similarly, for tools, we ask which ones enable the widest range of other things to be created, potentially even unanticipated ones. So the question almost always also starts to shift into one of "in what circumstances does it work?". Finally, even when a system does not work well, we may still learn something useful if there is enough promise that the concept might be made to work and we uncover information about what problems need to be overcome.

Overall, the knowledge we seek to elicit through an implementation tends to be rich and varied. Correspondingly, as described in the next section, the types of implementation approaches seen in typical practice also tend to take on a wide variety of forms and approaches (and none really dominates). There are many different implementation platforms that may be used, ranging from scripting or prototyping platforms not normally suitable for production use to "industrial strength" platforms of the same type that might be used for a final implementation. Similarly, implementations may consider only a very narrow range of function—only that which is new or what is strictly necessary to demonstrate the concept alone—or may include a richer set of functions necessary to make use of it in more realistic settings. In the end, to be sufficient, a proof-of-concept implementation needs to be complete enough to answer both the basic questions of "does it work (well enough, etc.)?" and any set of additional questions that we might wish to ask in an extended evaluation.

Types of Proof-of-Concept Implementations

Many proof-of-concept implementations take a form that can best be described as a *demonstration*. To succeed, that demonstration must illustrate the worth of the invention and in many cases motivate why it should be considered a success. Demonstrations fall along a rough scale of completeness or robustness. As used in the HCI research community, the presentation form of a demonstration is an indirect measure of its robustness, ordered below from the least to the most robust:

- Description in prose
- Presentation through photos (or screen dumps) showing the invention working
- Video showing the invention in use
- Live demonstration by the inventors
- Testing of properties with users
- Deployment to others to use independently

Presentation type works as a rough surrogate indicator because as we progress along this scale, more and more robustness or completeness is required to adequately present it (in part because the circumstances become less and less controlled or more open and arbitrary).

While higher levels of robustness or completeness clearly provide improved evidence about the quality of the invention, progression along this scale also involves dramatically increased levels of effort and resources. For example, deployment for widespread use can require a level of completeness nearly identical to a full product. (see Chap. Field Deployments: Knowing from Using in Context, this volume.) This often brings with it a need for development efforts that touch on many things not particularly relevant to evaluating the invention in question. Yet this extremely high level of effort may provide only a small increment in additional knowledge. In fact in the worst case, a high-end demonstration involving something like a deployment can even introduce enough confounds unrelated to the core invention that it actually obscures our understanding of it. For example, a deployment may fare very poorly with end users, but this might be due to factors completely unrelated to the worth of the core invention.

For example, suppose we have invented a way to help people who are deaf to find out about the content of ambient sounds in their environment (e.g., Matthews, Fong, Ho-Ching, & Mankoff, 2006). This piece of work, originally completed in 2004, depended on a human to transcribe audio that was shipped to them at the request of a participant who pressed a "What happened?" button on their mobile phone. At the time, technologies that would make this easy to implement today were not available: smartphones were just beginning to be available (but Android and the iPhone were not), Mechanical Turk was less than a year old, speech recognition could only function in constrained environments, and non-speech audio was not easily recognized. Our "deployment" lasted only a few weeks and required of users that they deal with cellular network wait times of up to 9 h and depend on a single human transcriber who was only available for a limited set of hours each day. From a technical perspective, *all of these barriers were peripheral to the invention itself.*

Our validation consisted of our proof-of-concept implementation and was (in this case) enhanced by some data on places and ways in which the technology was used by users who were willing to put up with the other difficulties. At the time, nothing similar existed, so the appropriate goal for the work was to answer the question "can we do this at all?". Our study also answered some questions about "what sounds need to be recognized to automate this?" (such as emotion, non-speech audio) and in the process answered some questions about "where might people use this?" though the last contribution was not strictly necessary for the work to make a technical contribution. In the six years since the work was published all but one (the recognition of non-speech audio) have been "solved." Thus, similar work done more recently has pushed much further on raising the ceiling for what can be done. An example is VizWiz (Bigham et al., 2010) that introduced a new way to use crowd workers to increase the speed of real-time image interpretation for the blind, and Legion Scribe (Lasecki, Miller, Kushalnagar, & Bigham, 2013), which made further advances to enable real-time captioning of videos.

However, from a technical HCI perspective, the value of the invention was clear (and publishable) irrespective of these difficulties.

As a result, it is critical to find an appropriate trade-off between robustness and completeness compared to the cost and effort necessary to create such an implementation. If we were to insist that each invention has the most robust implementation before we could trust its worth enough to build on it, progress in the field would be dramatically reduced—we would spend our time creating many fewer things and so decrease our ability to learn from, and build on, the previous efforts.

Alternatives to Proof-of-Concept Implementations

Although proof-of-concept implementations at some level are considered necessary as a basic validation, there are times when they are either not appropriate or not possible. For example, one less common way to make a contribution is to categorize or organize prior work in an area in a way that places it in a much more understandable light. This includes for example creating a useful taxonomy for a body of work, such as the design space of input devices put forth by Card and Mackinlay (1990). While this does not involve the creation of any new invention per se, it requires the creation of a conceptual framework of new organizing principles. Such a framework may highlight properties that have not been combined or identify areas that have not been explored. For example, our review of approaches to handling uncertainty in user input (such as touch screen input or gestural input) breaks uncertainty down into target uncertainty (where did the user click or what did he or she intend to interact with), recognition uncertainty (what interaction type is indicated) and segmentation uncertainty (where did an input begin and end) (Mankoff, Hudson, & Abowd, 2000a, 2000b). By viewing related work through the lens of different types of uncertainty, we can see that very few if any researchers have addressed segmentation uncertainty in the same depth that other forms of uncertainty have been addressed. Observations such as these can point to areas that are "ripe" for new work and thus make it easier to invent new things.

Another occasion when proof-of-concept implementations are less viable is when a concept requires something beyond the current state of the art to realize. While we might consider such concepts impractical and discard them, they can be very valuable contributions. For example, imagine an application that requires two problems to be solved (such as more accurate eye tracking in real-world contexts and more robust registration of the user's head position with the world). It may be possible to make progress in one area (more robust registration, say) while waiting for progress in the other. Similarly, we may want to demonstrate the high value in terms of unrealized applications of a currently unsolved problem as motivation for others to direct their attention and resources to solving it. Because of the value of being able to consider concepts seemingly beyond the present capability, the community has developed several approaches to learning about the properties of these concepts. These include *buying a time machine*, *Wizard of Oz* approaches, and *simulation*.

Buying a Time Machine

One approach to working beyond the state of the art is what is sometimes called *buying a time machine*. This approach involves spending a comparatively large sum of money or other resources—a sum too large to be justified for a real product of the same type—to get access now to technology that we can expect to be much more affordable and/or practical in the future. For example, we might be able to explore the capabilities of a future home vacuum-cleaning robot with very sophisticated vision processing by implementing the vision processing on a rented high-end supercomputer that communicates with the robot wirelessly. It is not currently practical to put a supercomputer in a vacuum cleaner, but the exponential growth of computing power described by Moore's law makes it reasonable to assume that the equivalent computing power will be available in a single-chip computer in the future.

Unfortunately, in the area of general-purpose computing, it is harder to *buy a time machine* today than it has been in the past. For example, in the middle of 1980s technical HCI researchers could employ what were then high-end workstations that performed 10 or even 100 times faster than typical consumer products of the era. This allowed them to explore the properties of systems that would not be widely practical for consumers for another 5–10 years. However, because of changes in the market for personal computers, it is not that easy to leap ahead of the "average" system today. On the other hand, advanced systems today are incredibly capable and diverse in comparison to past systems. Additionally, today's researchers may exploit graphic processing units (GPUs), create custom electronic circuits, or use (currently) more expensive fabrication techniques such as 3D printing to explore concepts. Each of these technologies allows us to make use of technologies that will likely be more practical and ubiquitous in the future but also currently comes at a cost in terms of requiring specialized skills or approaches.

Wizard of Oz Prototyping

Wizard of Oz prototyping involves simulating advanced capabilities by means of a hidden human who performs actions that a future system might be able to provide autonomously. This method was originally developed to explore user interface aspects of natural language understanding systems that could not yet be built in order to inform how such a system should be structured (Kelley, 1983, 1984). The Wizard of Oz approach clearly has some substantial advantages, both for exploring currently unattainable capabilities and simply for more rapidly and inexpensively simulating attainable ones. However, care must be taken to limit the wizard to an appropriate set of actions and to understand the effects that differences such as slower response times might have.

Simulation

A final way in which we might explore concepts that are impractical or impossible to build is to make use of simulation. This can take the form of simulating some or all of a system or of providing simulated rather than actual input data. A related set of techniques has recently emerged in the form of *crowdsourcing* (see Chap. Crowdsourcing in HCI Research, this volume), wherein large numbers of human workers recruited by services such as Amazon's Mechanical Turk can provide forms of *human computation* (simulating what otherwise might be computed by a machine). Interestingly, recent research shows that it may be possible not only to temporarily substitute human computation for future parts of a system but also to consider using crowdsourcing techniques as a part of a deployed system (Bernstein et al., 2010; Bernstein, Brandt, Miller, & Karger, 2011).

Secondary Forms of Validation

Beyond the central questions surrounding "(In what circumstances) does it work (well enough)?" there are a wide range of other criteria by which we can validate invention in HCI. These follow a set of properties that the community often sees as valuable.

Validations of Inventions Providing Direct Value for Human Needs

For inventions that are providing a direct contribution, we value creating an artifact that meets a stated human need. These needs are often met by creating a new capability or by speeding or otherwise improving a current capability. Perhaps the most common evaluation methods we see employed to demonstrate this are usability tests, human and machine performance tests, and what we will call expert judgment and the prima facie case. Although these are not universally appropriate, they are the most common in the literature.

Usability Tests

Because of the current and historical importance of usability and related properties as a central factor in the *practice* of HCI, usability tests of various sorts have been very widely used in HCI work and are the most recognizable of evaluation methods across the field. In fact the authors have frequently heard the assertion among students and other beginning HCI researchers that "you can't get a paper into CHI[1] without a user test!"

[1] *The ACM SIGCHI Conference on Human Factors in Computing Systems*, which is the largest HCI conference and seen by many as the most prestigious publication venue for HCI work.

This assertion is demonstrably false. An invention must be validated, but validation can take many forms. Even if a usability test shows that an invention is easy to use, it may not be very impactful. Its ability to be modified, extended, or applied to a different purpose may be much more important than its usability. Additionally, while user-centered methods may help with iterative design of a product, for the actual act of inventing—the conception of a new thing—usability tests offer relatively little assistance. However, usability testing (and other user-centered methods) does represent a bias of the community at large, particularly when results are going to be presented to, or evaluated by, a wide audience within our diverse field. This is likely true because they are one of the few evaluation methods with which every HCI researcher is sure to be familiar with.

On the other hand, usability tests are clearly appropriate when they match the properties of a research advance. Any research that puts forward an artifact or a system intended to provide improvements in usability, user experience, etc. clearly needs to present evidence that this is the case. There are a range of widely employed methods for doing this. Not all inventive research seeks to improve on user-centered properties. Indeed, it is critically important that we do not push for all or even most inventive research to aim mainly at these goals. If we were to do that, the field would suffer substantially because in early stages of work on a new type of artifact we must often first get past the questions such as "can we do this at all?" and "what capabilities are most important?" before considering whether something is useful/usable/desirable/etc.

Photo in the right is copyright ©1997 by Steven Feiner (used with permission). Photos in the left are (*top*) "New York Times on iPhone 3GS" by Robert Scoble, http://www.flickr.com/photos/scobleizer/4697192856, and (*bottom*) "Details of Google Glass" by Antonio Zugaldia, http://www.flickr.com/photos/azugaldia/7457645618/, both published under a Creative Commons Attribution 2.0 Generic License

In short, as illustrated in the figure above, it is often necessary to pass through decidedly non-usable stages to create the technology necessary to make something that in the end delivers a great user experience.

Human Performance Tests

Another very widely used class of evaluation methods involves measuring the performance of typical users on some set of tasks. These tests are most applicable when goals for results revolve around a small set of well-defined tasks. Work in interaction techniques is one of the few areas where this type of validation is consistently appropriate. Because some interactive tasks recur frequently, this is also one of the few areas where at least some consistent and reusable measures have emerged. In particular, measurement of pointing performance within a Fitts' law framework (e.g., determining Fitts' law coefficients for devices and interaction techniques) is common because pointing and selection tasks are fundamental to many interactive systems (MacKenzie, 1992; Wobbrock, Cutrell, Harada, & MacKenzie, 2008). Similarly measures of efficiency in text entry such as keystrokes per character (Mackenzie, 2002) have become well developed because text entry is a common task that has received considerable inventive attention.

One danger in using this kind of evaluation is that human performance tests are easiest to apply to narrow and well-defined tasks and generally seen as most valid when they are carefully controlled. Unfortunately, this leads away from the values of wide applicability of results (e.g., an invention useful for a wide range of tasks) and so can be in conflict with other properties of interest for inventive HCI research. Instead of looking for statistically significant improvements, it is important to focus on practical significance (effect size), and unfortunately there are no simple or widely accepted criteria for that. So while human performance tests are widely accepted and understood by the community, without care they can be much less useful than their popularity might indicate. (See Chapter on Experimental Research in HCI, this volume.)

Machine Performance Tests

Tests can also be done to measure the performance of an artifact or an algorithm rather than the person who uses it. These can be very practical in providing information about the technical performance of a result such as expected speed, storage usage, and power consumption. These measures resemble the validation measures commonly used in other domains such as systems research in computer science. It is often considered valid to *simulate* use across a range of conditions to generate such measures. Although this may be indirect and lack real-world validity, such tests of technical performance can in turn point to likely effects on end users such as expected response times or battery life of a device. Similarly, tests could indicate

properties such as "*runs at frame rate*[2]" that may indicate that the part of the system being tested is unlikely to be a bottleneck in overall performance, thus telling the researcher that it may be appropriate to turn to improving other parts of the system in the future.

Expert Judgment and the Prima Facie Case

Properties such as innovation and inspiration are of substantial value for many research results. Opening new areas others had not considered before and providing a motivated basis for others to build within them are central to progress within the community. However, these factors are extremely hard if not impossible to measure in any standardized way. For these important but more nebulous properties we most typically must rely on what amounts to expert opinion—whether the result impresses other researchers experienced in the area. This is often done with demonstrations and/or scenarios that are intended to present a prima facie case for innovation and/or inspiration. In essence these are intended to elicit a reaction of "Wow, that's cool!" from experts who know the area well and can informally compare it to the state of the art. Such a reaction is a rapid and informal but an experienced-based assessment that the work has important properties such as advancing the state of the art, opening up new possibilities, or taking a fresh approach to an established problem. For example, inventions may open a new area that had not been conceived of before (such as inspiring large numbers of people to do small bits of useful work by playing a game, see von Ahn & Dabbish, 2008) or take a substantially different approach to a problem that many others have worked on (such as recognizing activities in a home by listening to water pipes and electrical noise in the basement (Fogarty, Au, & Hudson, 2006; Patel et al., 2007) or identifying people based on recognizing their shoes, see Augsten et al., 2010).

Clearly this type of validation has problems. It is very dependent on the subjective opinion of experts (most notably reviewers of papers seeking to publish the results) and as such is not very reliable or repeatable. Applying validations of this form to activities of discovery would normally be unacceptable. But in activities of invention where we usually must deal with the uncontrolled complexity of the world, and often seek the widest circumstances for applicability, we are almost never able to know everything we need to know with certainty. As a result follow-on work tends not to make strong assumptions about the applicability of past validation to current circumstances. This means that the uncertainty associated with this type of validation can be more acceptable and less damaging if it turns out to be wrong.

Validation of this form is seen fairly widely in practice—things are valued based on informal assessment of their level of innovation and inspiration by experts, in colloquial terms things treated as having value in part because "they seem cool" to

[2] This is the rate of display refresh (which is typically 50 or 60 times per second in order to avoid perceived flicker). This rate is of particular interest because even if internal updates to visual material occur faster than this, they will still never be presented to the user any faster than this.

those with experience in similar work. However, the uncertain properties of this approach make reliance on this type of validation alone a rather risky and unpredictable approach, both for the inventor seeking acceptance of a single invention and the field in making progress overall. To overcome this, most inventions that are validated in this way often seek to provide additional forms of validation (starting with proof-of-concept implementations).

Validations of Tools That Have Indirect Impact on Human Needs

We now consider validation methods for our second set of contributions: those that provide indirect value—that contribute to something that enables or promotes an eventual practical impact rather than providing it directly. For these properties, a rather different set of approaches to validation are appropriate.

One of the most important forms of validation for enabling tools is the use of examples of things that can be built with the tool that demonstrate certain desirable properties of the tool. These can include demonstrations of lower threshold, higher ceiling, breadth of coverage of a desirable design space, increased automation, and good abstractions or extensibility, discussed in more detail below. For inventions involving base capabilities (which are often aimed at overcoming specific roadblocks or limitations of prior work) machine performance tests and in some cases illustration of a prima facie case may be useful.

Threshold, Ceiling, and Breadth of Coverage

A primary example of how inventions help researchers make useful things is *improvements in threshold or ceiling effects* (Myers, Hudson, & Pausch, 2000). (Threshold effects relate to the ease with which simple things can be accomplished and/or novice users can get started, whereas ceiling effects are related to the limitations of a tool or a system for creation of complex or unexpected things.) Validating a low threshold for a tool is often done with a demonstration where the inventor illustrates that something, which in other tools requires considerable work, can be created in their tool easily. For example, the inventor may demonstrate how something can be built in a small number of steps or using a small amount of specification code. Validating a high ceiling is most typically done via a demonstration wherein the inventor shows that one or more sophisticated or complex things—often things that are out of the practical reach of other tools—can be created with their tool. Unfortunately, low threshold tools often tend to impose a low ceiling, and high ceiling tools often come with a high threshold. Consequently, finding ways to ensure both low threshold and high ceiling in one tool is highly valued. Illustration of breadth of coverage is often provided by demonstrating a *spread of examples*—that is, a set of examples that are very different and that span a large(r) space within the set of possible results.

These types of validation all involve creating examples with the tool. Note that the validation is about creation of the examples, but the full properties of the resulting examples are usually not the central issue. So validations that address the properties of the examples themselves are generally not appropriate. For example, performing a usability test on an example built with a tool would likely tell us almost nothing about the tool—many different things might be built with any good tool, and the usability of those things is at least as much a reflection of the designer using tool as it is a property of the tool. Instead the simplicity of creation (for threshold), the power or complexity (for ceiling), or the variety (for breadth of coverage) of the examples is what is critical.

As with other sorts of inventions, machine performance tests may be valuable for enabling technologies. For example, in the case of increased automation it can be appropriate to use performance tests to show that the results are comparable to what is created by previous non-automated methods. Similarly, it may be valuable to demonstrate that the abstractions employed work as the use of the tool scales up. This can be proven in part using simulation, but description and logic may also play a role.

Presentation of Good Abstractions

Like the other validations appropriate for tools and systems, a typical validation for good abstractions is through a set of illustrative examples. To illustrate extensibility, these examples are often similar to breadth of coverage examples, in that illustrating a spread of applicability is useful. For illustrating improved understanding, or ease of application, sets of examples are often similar to those used to illustrate improvement in floor or ceiling effects. While at times this is validated by having developers actually use a toolkit and exploring the details of what they built (see below) this is in many cases a prohibitively expensive way to validate, and it is often considered sufficient to describe abstractions and clearly contrast them with prior alternatives.

Usability for Developers

In some cases, usability tests may be carried out with enabling tools. However these tests need to focus on the *developers* who may be using the tool to create applications, not on the *end users* of the applications created. The number of confounds affecting our ability to evaluate whether a tool engenders usable applications from an end-user perspective is enormous, and the usability of applications is often not the primary value of the tool and should not be the central focus of validation efforts.

Some evaluation of developers working with tools has focused on what abstractions they make use of. When a tool is sufficiently far along to have a large developer community, it can also be interesting to look at metrics such as what types of applications were built with the tool and how the tool was extended. This begins to

resemble studies of programmers, programming, and open-source communities. However the cost of bringing a tool this far along may be prohibitive especially when compared to the benefits for invention. Further, because of the high number of confounding factors that may be outside the scope of the tool advance being presented, this type of validation can actually be quite "noisy." In particular, it is very difficult to separate the effects arising from extraneous usability issues in tool interfaces being compared from those related to the core concepts of the tools.

Summary

At this point we must step back and note that the primary form of evaluation for enabling technologies is to build key parts of the technology (proof-of-concept creation). As outlined above, after this primary step it is typical to consider additional validation that highlights the specific goals of the work, that is, to describe the abstractions it employs clearly or to build examples that demonstrate the capabilities of the technology. While there are some secondary evaluations that involve (end) user studies, these are rarely employed.

Summary and Conclusion

In this chapter we have considered the nature of technical work in HCI. To do this we have first situated the work in a broad framework that contrasts its inventive character with one of the other dominant bodies of activities within HCI: those of discovery. This high-level characterization of the work is useful because it allows us to see fundamental differences in the nature of the two kinds of work. These in turn lead to very different values and methods that have evolved to suit each type of work. For example, we conclude that the specifics of methods used in activities of discovery are extremely important—so much so that results are not really understandable in isolation from the methods used to reach them, and so they really become part of the results themselves. In contrast, for activities of invention, the use of one method versus another is much more fluid and less fundamental. Instead, the application of the invention, as demonstrated through a proof-of-concept implementation of the thing invented, is a crucial component of the result.

Using this overall conceptual framework we then consider inventive HCI work itself. We characterize two broad categories of contributions: direct and indirect—where direct contributions directly contribute to meeting some human need, while indirect contributions serve as enablers for later work that meets some human need.

We then go on to characterize the tasks of inventive work in HCI. These tasks include concept creation and validation of concepts. However, we note that one form of validation—the building of proof-of-concept implementations—is more fundamental than other forms. Because it addresses the basic issue of "does it work?"

a proof-of-concept implementation represents a prerequisite for other validation of the work. Because of its special nature it is the normal practice in technical HCI to give proof-of-concept implementations separate and stronger consideration than other forms of validation. As a result, we conclude that technical HCI work should be considered in three parts: concept creation, validation through proof-of-concept implementation, and other validation. The creation of a proof-of-concept implementation (which may need to be quite complex in some cases, as with a toolkit) is a key point of difference with other forms of HCI: Technical HCI is about making things that work, and the work of technical HCI is not done until the validation inherent in an implementation (at a minimum) is complete.

We explore each of these three parts separately. There are few specific methods that one can expect to provide consistently positive outcomes for concept generation. However, we do consider several general strategies for going about the work. These include needs-first and technology-first approaches. We also point to some advantages for technology-first approaches, even though they have developed a somewhat tarnished reputation within the HCI research community. We then consider validation through proof-of-concept implementations by looking at why they are so critical and central. We elucidate the questions that they can address and highlight the diminishing returns inherent in making a prototype complete and robust.

Finally, we consider a range of different forms of secondary validation that can be useful. We characterize a range of different measures we might be interested in and then consider an equally wide range of techniques that can be applied to provide information in those areas. We emphasize again that we must consider a trade-off between the level of knowledge to be gained and the costs of these evaluations and point to places where our community has not always succeeded in choosing the best evaluation methods.

It is typical that technical researchers learn these methods and ideas through osmosis—few courses teach approaches to validating technical work or concept creation in the way that study design and analysis are taught, for example. Instead, technical education programs tend to give researchers the necessary knowledge base from which to invent (how to program, how to use machine learning, how to build circuits, and so on) and hope that with that knowledge, the examples of those who came before (and the guidance of mentors), and a good dose of creativity the novice research will create good results. This chapter has set out to rectify some of those gaps by putting common practice, and the rationale behind it, into words.

Exercises

1. Compare and contrasts technical HCI research with research through design.
2. Where do the ideas come from for technical HCI research? What is the problem that researchers are solving?

References

Appert, C., & Beaudouin-Lafon, M. (2008). SwingStates: adding state machines to Java and the Swing toolkit. *Software: Practice and Experience, 38*(11), 1149–1182.

Augsten, T., Kaefer, K., Meusel, R., Fetzer, C., Kanitz, D., Stoff, T., et al. (2010). Multitoe: high-precision interaction with back-projected floors based on high-resolution multi-touch input. *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology (UIST' 10)*, (pp. 209–218).

Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., et al. (2010). Soylent: a word processor with a crowd inside. *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology (UIST' 10)*, (pp. 313–322).

Bernstein, M. S., Brandt, J., Miller, R. C., & Karger, D. R. (2011). Crowds in two seconds: enabling real-time crowd-powered interfaces. *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST' 11)*, (pp. 33–42).

Bigham, J. P., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R. C., et al. (2010) VizWiz: nearly real-time answers to visual questions. *UIST 2010* (pp. 333–342).

Buxton, W., Lamb, M. R., Sherman, D., & Smith, K. C. (1983). Towards a comprehensive user interface management system. *SIGGRAPH Computer Graphics, 17*(3), 35–42.

Card, S. K., English, W. K., & Burr, B. J. (1978). Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics, 21*, 601–613.

Card, S. K., Mackinlay, J. D., & Robertson, G. G. (1990). The design space of input devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI' 90)*, (pp.117–124).

Cardelli, L. (1988). Building user interfaces by direct manipulation. *Proceedings of the 1st Annual ACM SIGGRAPH Symposium on User Interface Software (UIST' 88)*, (pp. 152–166).

Jayant, C., Ji, H., White, S., & Bigham, J. P. (2011). Supporting blind photography. In *The proceedings of the 13th international ACM SIGACCESS conference on computers and accessibility* (pp. 203–210). New York, NY: ACM.

Cohn, G., Morris, D., Patel, S. N., & Tan, D. S. (2011). Your noise is my command: sensing gestures using the body as an antenna. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*, (pp. 791–800).

Engelbart, C., & English, W. K. (1968). *AFIPS Conference Proceedings of the 1968 Fall Joint Computer Conference,* San Francisco, CA, December 1968, (Vol. 33, pp. 395–410)

English, W. K., Engelbart, D. C., & Berman, M. L. (1967). Display-Selection Techniques for Text Manipulation. *IEEE Transactions on Human Factors in Electronics, HFE-8*(1), 5–15.

Feiner, S., MacIntyre, B., Hollerer, T., & Webster, A. (1997). A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. *Proceedings of the 1st IEEE International Symposium on Wearable Computers (ISWC' 97)*, (pp. 74–81).

Fogarty, J., Au, C., & Hudson, S.E. (2006). Sensing from the basement: A feasibility study of unobtrusive and low-cost home activity recognition. *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST' 06)*, (pp. 91–100).

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology, 47*, 381–391. Reprinted in Journal of Experimental Psychology: General, 1992, 121(3), 262–269.

Grossman, T., & Balakrishnan, R. (2005a). The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI' 05)*, (pp. 281–290).

Grossman, T., & Balakrishnan, R. (2005b). A probabilistic approach to modeling two-dimensional pointing. *ACM Transactions on Computer–Human Interaction, 12*(3), 435–459.

Grossman, T., Kong, T., & Balakrishnan, R. (2007). Modeling pointing at targets of arbitrary shapes. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI' 07)*, (pp.463–472).

Gupta, S., Chen, K. Y., Reynolds, M. S., & Patel, S. N. (2011). LightWave: Using compact fluorescent lights as sensors. *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp' 11)*, (pp. 65–74).

Harrison, B. L., Fishkin, K. P., Gujar, A., Mochon, C., & Want, R. (1998). Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI' 98)*, (pp. 17–24).

Hinckley, K., Pierce, J., Sinclair, M., & Horvitz, E. (2000). Sensing techniques for mobile interaction. *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST' 00)*, (pp. 91–100).

Hudson, S. E., Mankoff, J., & Smith, I. (2005). Extensible input handling in the subArctic toolkit. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI' 05)*, (pp. 381–390).

Hullot, J. M. (1986). SOS Interface. *Proceedings of the 3rd Workshop on Object Oriented Programming, Paris, France*, Jan. 1986.

Ishii, H., Kobayashi, M., & Arita, K. (1994). Iterative design of seamless collaboration media. *Communications of the ACM, 37*(8), 83–97.

Jacob, R. J. K. (1986). A specification language for direct-manipulation user interfaces. *ACM Transactions on Graphics, 5*(4), 283–317.

Kelley, J. F. (1983). *Natural language and computers: Six empirical steps for writing an easy-to-use computer application*. Doctoral dissertation, The Johns Hopkins University, Maryland.

Kelley, J. F. (1984). An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems, 2*(1), 26–41.

Khan, A., Fitzmaurice, G., Almeida, D., Burtnyk, N., & Kurtenbach, G. (2004). A remote control interface for large displays. *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST' 04)*, (pp. 127–136).

Lasecki, W. S., Miller, C. D., Kushalnagar, R. S., Bigham, J. P. (2013). *Legion scribe: real-time captioning by the non-experts. W4A 2013* (pp. 22).

MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human–Computer Interaction, 7*, 91–139.

MacKenzie, I. S. (2002). KSPC (keystrokes per character) as a characteristic of text entry techniques. In Fabio Paternò (Ed.), *Proceedings of the 4th international symposium on mobile human–computer interaction* (*Mobile HCI' 02*), (pp. 195–210).

Mankoff, J., Hudson, S. E., Abowd, G. D. (2000a). Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. *Proceedings of CHI 2000*, (pp. 368–375).

Mankoff, J., Hudson, S. E., & Abowd, G. D. (2000b). Interaction techniques for ambiguity resolution in recognition-based interfaces. *Proceedings of UIST 2000*, (pp. 11–20).

Matthews, T., Fong, J., Ho-Ching, F. W.-L., & Mankoff, J. (2006). Evaluating visualizations of non-speech sounds for the deaf. *Behavior and Information Technology, 25*(4), 333–351.

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5*(4), 115–133.

Meyer, D. E., Smith, K. J. E., Kornblum, S., Abrams, R. A., & Wright, C. E. (1990). Speed-accuracy tradeoffs in aimed movements: Toward a theory of rapid voluntary action. In M. Jeanerod (Ed.), *Attention and performance XIII* (pp. 173–226). Erlbaum: Hillsdale, NJ.

Myer, B. A. (1990). A new model for handling input. *ACM Transactions on Information Systems, 8*(3), 289–320.

Myers, B. A., McDaniel, R. G., Miller, R. C., Ferrency, A. S., Faulring, A., Kyle, B. D., et al. (1997). The Amulet environment: New models for effective user interface software development. *IEEE Transactions on Software Engineering, 23*(6), 347–365.

Myers, B., Hudson, S. E., & Pausch, R. (2000). Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction, 7*(1), 3–28.

Mynatt, E.D., Essa, I., & Rogers, W. (2000). Increasing the opportunities for aging in place. *Proceedings on the 2000 Conference on Universal Usability (CUU' 00)* (pp. 65–71).

Mynatt, E. D., Rowan, J., Craighill, S., & Jacobs, A. (2001). Digital family portraits: supporting peace of mind for extended family members. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI' 01)*, (pp. 333–340).

Newman, W.M. (1968). A system for interactive graphical programming. *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference (AFIPS' 68 (Spring))* (pp. 47–54).

Patel, S. N., Robertson, T., Kientz, J. A., Reynolds, M. S., & Abowd, G. D. (2007). At the flick of a switch: Detecting and classifying unique electrical events on the residential power line. *Proceedings of the 9th International Conference on Ubiquitous Computing (Ubicomp' 07)*, (pp. 271–288).

Schwarz, J., Harrison, C., Hudson, S., & Mankoff, J. (2010a). Cord input: An intuitive, high-accuracy, multi-degree-of-freedom input method for mobile devices. *Proceedings of CHI' 10*, (pp. 1657–1660).

Schwarz, J., Hudson, S., & Mankoff, J. (2010b) A robust and flexible framework for handling inputs with uncertainty. *Proceedings of UIST' 10*, (pp. 47–56).

Schwarz, J., Mankoff, J., & Hudson, S. (2011). Monte Carlo methods for managing interactive state, action and feedback under uncertainty. *Proceedings of the 24th annual ACM symposium on user interface software and technology (UIST' 11)*, (pp. 235–244). New York, NY: ACM.

Shoemaker, G., Tang, A., & Booth, K. S. (2007). Shadow reaching: a new perspective on interaction for large displays. *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST' 07)*, (pp. 53–56).

Sutherland, I. E. (1963). SketchPad: A man–machine graphical communication system. *AFIPS Conference Proceedings, 23*, 323–328.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1*, 511–518.

von Ahn, L., & Dabbish, L. (2008). Designing games with a purpose. *Communications of the ACM, 51*(8), 58–67.

Wasserman, A. I. (1985). Extending state transition diagrams for the specification of human-computer interaction. *IEEE Transactions on Software Engineering, 11*(8), 699–713.

Wobbrock, J. O, Cutrell, E., Harada, S., & MacKenzie, I. S. (2008). An error model for pointing based on Fitts' law. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI' 08)*, (pp. 1613–1622).