# Benevolent Deception in Human Computer Interaction

**Eytan Adar**
University of Michigan
Ann Arbor, MI 48109
eadar@umich.edu

**Desney S. Tan**
Microsoft Research
Redmond, WA 98052
desney@microsoft.com

**Jaime Teevan**
Microsoft Research
Redmond, WA 98052
teevan@microsoft.com

**ABSTRACT**
Though it has been asserted that "good design is honest," [42] deception exists throughout human-computer interaction research and practice. Because of the stigma associated with deception—in many cases rightfully so—the research community has focused its energy on eradicating malicious deception, and ignored instances in which deception is positively employed. In this paper we present the notion of *benevolent deception*, deception aimed at benefitting the user as well as the developer. We frame our discussion using a criminology-inspired model and ground components in various examples. We assert that this provides us with a set of tools and principles that not only helps us with system and interface design, but that opens new research areas. After all, as Cockton claims in his 2004 paper "Value-Centered HCI" [13], "Traditional disciplines have delivered truth. The goal of HCI is to deliver value."

**Author Keywords**
Benevolent deception; criminology; design principles.

**ACM Classification Keywords**
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

**INTRODUCTION**
Users generally trust computer interfaces to accurately reflect system state. Reflecting that state dishonestly—through deception—is viewed negatively by users, rejected by designers, and largely ignored in HCI research. Many believe outright deception should not exist in good design. For example, many design guidelines assert: "Do not lie to your users" (e.g., [40, 45]) Misleading interfaces are usually attributed to bugs or poor design. However, in reality, deceit often occurs both in practice and in research. We contend that deception often helps rather than harms the user, a form we term *benevolent deception*. However, the overloading of "deception" as entirely negative coupled with the lack of research on the topic, makes the application of deception as a design pattern problematic and ad hoc.


*CHI 2013*, April 27–May 2, 2013, Paris, France.


Benevolent deception is ubiquitous in real-world system designs, although it is rarely described in such terms. One example of benevolent deception can be seen in a robotic physical therapy system to help people regain movement following a stroke [8]. Here, the robot therapist provides stroke patients with visual feedback on the amount of force they exert. Patients often have self-imposed limits, believing, for example, that they can only exert a certain amount of force. The system helps patients overcome their perceptive limits by underreporting the amount of force the patient actually exerts and encouraging additional force.

The line between malevolent and benevolent deception is fuzzy when the beneficiary of the deception is ambiguous. For example, take the case of deception in phone systems to mask disruptive failure modes: The connection of two individuals over a phone line is managed by an enormous specialized piece of hardware known as an Electronic Switching System (ESS). The first such system, the 1ESS, was designed to provide reliable phone communication, but given the restrictions of early 1960s hardware, it sometimes had unavoidable, though rare, failures. Although the 1ESS knew when it failed, it was designed to connect the caller to the wrong person rather than react to the error in a more disruptive way (e.g., disconnect, provide some message, etc.). The caller, thinking that she had simply misdialed, would hang up and try again: disruption decreased, and the illusion of an infallible phone system preserved [41].

A further example of benevolent deception are the "placebo buttons" that allow users to feel as though they have control over their environment when they actually do not. Crosswalk buttons, elevator buttons, and thermostats [33, 47] often provide no functionality beyond making their users feel as though they can affect their environment. Some of these buttons go far to provide the illusion of control; non-working thermostat buttons, for example, are sometimes designed to hiss when pressed [2]. In addition to providing the feeling of control, placebo buttons can signal the existence of a feature to the user. Non-working crosswalk buttons, for example, clearly convey to a pedestrian that a crosswalk exists.

As is the case with the 1ESS and placebo buttons, deception sometimes benefits the system designer, service provider, or business owner. However, this does not invalidate the fact that it might also help meet user needs. We believe that by not acknowledging that there is deception, and, more critically, that a line between beneficial and harmful decep-

tions might exist, research in the area is difficult to pursue—to the detriment of academics and practitioners alike.

Whether intentional or not, implicit or explicit, acknowledged or not, benevolent deceit exists in HCI. Nonetheless, little is known about the motivation, mechanisms, detectability, effectiveness, successes, failures, and ethics of this type of deception. Researchers have tiptoed around this taboo topic, concentrating instead on malevolent deception (e.g., malware or malicious software [14,17]) and unobjectionable forms of deception described using entertainment metaphors (e.g., magic or theater [32,54]). This limited view of deception does not capture its variety or ubiquity.

As we will see, one of the underlying reason for the ubiquity of deception is that it can fill the many of the gaps and tensions that emerge with different design concerns (e.g., the good of the individual versus the good of the group), design goals (e.g., conflicting principles), or systems states (e.g., desired system performance versus actual system performance). In any situation where a poor fit exists between desire (e.g., the mental model or user expectations) and reality (e.g., the system itself) there is an opportunity to employ deception. This gap—which is extremely common—both motivates and enables the deception.

Our goal in this paper is to provide an overview of the space—a working definition for deception—and to provide a framework for future research. We begin by defining deceit in HCI. We look at related metaphors, expanding them to include instances in which the user does not willingly participate in the deception. We then present a model of deceit framed around the *motive*, *means*, and *opportunity* of benevolent deception. We integrate possible research directions throughout and conclude with a discussion of a possible model for the application of deception.

## DEFINING DECEIT

Deceit is generally regarded as manipulation of the truth either by hiding truthful information or showing false information. *Deception* is an act of deceit with implied intent (e.g., telling the user the web page is 70% loaded when we know that this is not the case). On the other hand, *deceptive(ness)* does not require intent (e.g., telling the user that the web page is *absolutely* 70% loaded based on some estimate with high error margins). Though this distinction is important as it speaks to motive, deceit exists with or without intent. In fact, when deciding whether an advertisement is illegal (false), the FCC only considers the deceptiveness of the message irrespective of intent. That said, proving motive/intent in a design is also a very convincing argument for conviction. There is a notable difference between unintentional bugs, errors, or bad metaphors and ones that have been carefully designed for specific purposes.

It is further useful to distinguish between deceit that affects behavior directly or indirectly (by modifying the user's mental model). A test of this impact is whether a user would behave differently if they knew the truth. Though the line is fuzzy, this distinction allows us to separate abstraction from deception. Applying this thought experiment to deception interfaces, one might see that the 1ESS user who was connected to the wrong number might begin to blame and mistrust the system, the rehabilitation user may recalibrate what is on the screen and again be unwilling to push themselves, and so on. Though the line is fuzzy, this distinction allows us to separate abstraction (or simplification), in which user behavior is largely unchanged, from deception, in which it often is changed.

Deceit occurs when

1. an explicit or implicit claim, omission of information, or system action,
2. mediated by user perception, attention, comprehension, prior knowledge, beliefs, or other cognitive activity,
3. creates a belief about a system or one of its attributes,
4. which is demonstrably false or unsubstantiated as true,
5. where it is likely that the belief will affect behavior,
6. of a substantial percentage of users.

**Figure 1. A definition of deceit, based on [44].**

Building on behavioral and legal definitions introduced in earlier work [44] that deal with deceptive advertising, we put forth a working definition of deceit as it applies to HCI work in the Figure 1. Points 5 & 6, on substantial effect, are perhaps the most controversial, and are purposefully left ambiguous. How behavior is affected and what "substantial" means are left open, as there is likely no answer that works in every situation. A deceptive interface that causes physical harm in 1% of the user population may have a substantial effect, whereas an idempotent interface with a button that misleads significantly more users into clicking twice may not pass the substantial test.

In addition to intent, there are many other ontologies of deceit. Bell and Whaley [4] identify two main types of deception—*hiding* and *showing*—which roughly correspond to masking characteristics of the truth or generating false information (both in the service of occluding the truth). These forms of deception represent atomic, abstract notions of deceit that we refine in our discussion below. Related to the hiding/showing dichotomy is the split between silent (a deceptive omission) versus expressed deception. *Lying*, as a special class, is generally considered to be a verbal form of deception [5]. Because HCI need not involve a verbal element, we expand the notion of the "lie" to include nonverbal communication between humans and computers.

## DECEPTION IN HCI

Some in the HCI community [7, 24] have studied human-to-human deception in computer-mediated environments. For example, people may hide their presence in instant messaging clients to avoid unwanted interruptions, or auto-respond to messages at pre-programmed times to appear online when they are not. Research in this area has targeted the extent to which such deception happens and has ex-

plored features that enable this type of deception. While research into human-to-human deception has advanced our understanding of deception in general, it has largely focused on communication behavior that pre-exists, and persists through, computer-mediated-communication.

Most of the systematic research in the academic HCI community focuses on *malevolent deception*, or deception intended to benefit the system owner at the expense of the user [14]. Such research frames deception negatively, and focuses on detection and eradication of malicious or evil interfaces (e.g., *dark-patterns* [17]). Such patterns include ethically dubious techniques of using purposefully confusing language to encourage the addition of items to a shopping cart or hiding unsubscribe functionality. Many forms of malevolent deception, such as phishing and other fraud, is in clear violation of criminal law.

Deceptive practices that are considered harmful by legal organizations are, for generally good reasons, considered harmful by designers as well. A possible exception to the negative frame for malevolent deception is in contexts where an obvious adversary exists or for military or security purposes. For example, honeypot servers look like unprotected servers, but are meant to trap "hackers" by deceiving file sharing applications into believing they are disconnected. In this space, user harm is often treated as acceptable.

It can be difficult to draw the line between malevolent and benevolent deception. We frame the distinction from the end-user's perspective: if the end-user would prefer an experience based on the deceptive interface over the experience based on the "honest" one, we consider the deception benevolent. Note that this includes situations in which both the system designer and the end-user benefit from the deception, something economists sometimes term a *Pareto white lie* [19]. Arguably, all benevolent deceptions that improve user experience benefit the interface creator as well; otherwise, we might use the term *altruistic deception* [19].

The limited HCI research that has been conducted on benevolent deception has focused on the use of magic [54], cinema and theater [32] as instructional metaphors for HCI design. By using deception for the purpose of entertainment, playfulness, and delight, it becomes acceptable. This work has connected these art forms, in which illusion, immersion, and the drive to shape reality dominate, to situations in which there is willing suspension of disbelief on the part of the user, or at least willing pursuit of acceptable mental models. Similar lessons have been drawn from architecture. For example, theme parks and casinos [20, 34, 37] are designed specifically to utilize illusions that manipulate users' perception of reality, entertainment, and participation in the experience. In this paper, we will describe a number of deceits in HCI systems that have parallel designs to magic, theater, and architecture. However, not all benevolent deceit can be framed in terms of creating delight for the user, as it is often used for other purposes as well, such as to mask uncomfortable failures or align the interests of parties with different needs.

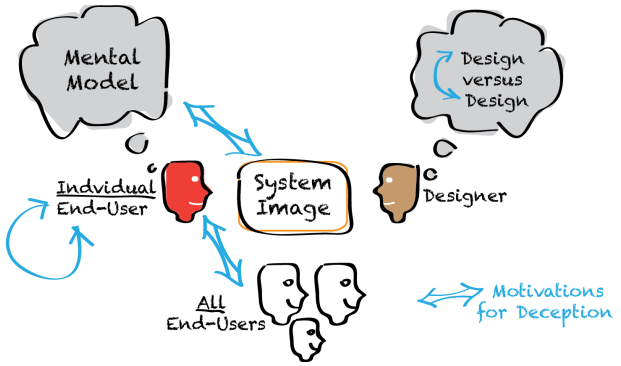


**Figure 2: Motives for Deception**

We propose a model of deception based on criminology, couching it in terms of *motive* (why it happens), *means* (how to do it), and *opportunity* (when it works). We view the framing of HCI deceit as a crime attractive in that it allows us to break apart the deception in a systematic way. Doing so allows us to contrast real crimes (the malevolent deceptions) from deceptive design decisions that aid the user. By framing deceit in these terms, we are better able to break apart the tensions in design that motivate and enable benevolent deception. We look more closely at each below.

## THE MOTIVE

We begin by looking at the motive behind benevolent deception in HCI, to understand why it happens and where best to use it. There are many tensions in user interfaces that emerge in the balancing that occurs among system capabilities, implementation, end-user expectations, values, and business demands. Any successful design, truthful or not, tries to find the sweet spot that balances these while providing value to the end-user. It is in the attempt to bridge the gaps among different demands that benevolent deception exists. For example, if either the system image or user's mental model [39] are not aligned with the design, the designer may resort to deception. This appears especially true when designs must create balance between maximizing the utility to the user and maximizing the benefit to the system's owner, while minimizing the risk to both parties.

We broadly divide the gap in demands into four types: 1) where there is a gap between the user's mental model and the underlying system model; 2) where the needs of an individual must be balanced with the needs of the group; 3) where a person must be protected from oneself; and 4) where meeting different conflicting design goals requires making compromises. These gaps, illustrated in Figure 2, are discussed farther below. We then look more closely at instances in which the motivation for the deception is not merely to benefit the end user but also the designer.

### Mental Model vs. System Image

The most common motivation for deception in HCI is the tension between what the user expects of a system and what the system can actually do, by any metric that can be applied to measure function (system uptime, latency, entertainment measures, security, etc.)

When the underlying system suffers from performance issues, interface designers can hide this fact from the end-users through deception. The phone call routing system described in the introduction (1ESS) illustrates how failure in connecting two phone users is hidden by deceiving the callers into believing they had misdialed [41]. Netflix will invisibly switch from a personalized recommender system (a costly system based on personalized suggestions) to a simpler one based on popular movies when its servers fail or are overwhelmed. By maintaining the same visual experience, the user is unaware of the failure [12]. The designers of these systems prefer the deception to the disruption of the user's experience. That said, whether the end-user truly appreciates these decisions is something that can and should be tested; if the deception does not ultimately benefit the end-user, it crosses from benevolent to malevolent.

Deceptions can also hide uncertainty. Sometimes the system designer simply does not know the system state (e.g., how long something will take). Rather than putting the burden of this uncertainty on the user (by saying, for example, that "this operation will take between 10 and 20 minutes"), the designer may resort to estimating the time and projecting it back to the user as absolute truth (saying instead, that "this operation will take 15 minutes.")

In other situations the deception exists to guarantee a certain level of pleasure and entertainment. For example, computer games employ a notion of *artificial stupidity* [55], in which the performance of a computer opponent is degraded to maximize the human player's experience. A computer-based player might hit a digital pool ball to leave the human player an opportunity to score or may leave an advantageous move open in chess. This is a more complex and less accurate way to vary the computer opponent's skill level than are other options (e.g., varying the accuracy of the artificial pool player or the depth searched in the min-max tree), but it increases people's enjoyment of the game.

Deception can also be used to increase people's level of comfort with an interface. Heightening an interface's anthropomorphic characteristics can lower the end-user's discomfort (e.g., by having a robot enact stylized human behaviors such as thinking [51]) and can increase the interface's credibility [15]. Such designs are purely artificial and often have no other purpose than to deceive the end-user. However, they can also help the user act in more predictable ways. For example, many speech and natural language systems benefit (in increased accuracy) from having the users believe they are speaking to a fellow human being because they speak more naturally [3,16].

Finally, as many systems shift from mechanical to digital, maintain consistency requires deception. For example, an electric vehicle or electronic shutter camera may employ artificial noise reminiscent of that generated by the mechanical system to remain consistent with the end-users expectations. This form of path dependence or backwards compatibility is frequently cited as a reason for skeuomorphic design elements [56], though they also exist for stylistic reasons or to drive the feeling of comfort in the known.

### Individual vs. Group

In systems in which there are many end-users, designers may employ deception to balance the needs of the individual with those of the group. For example, with limited resources and a line of waiting end-users, deception may be used to move people through the line without negatively affecting the individual's experience. Disney does this by employing techniques to provide the sensation of control in virtual reality spaces while still nudging individuals through a constrained path: producing salient features (called "weenies" in Disney-speak) such as far-off battles to drive motion in a certain direction and employing virtual characters to bump a user along (while appearing accidental) [36].

Some systems used by groups have explicit security requirements, as is the case with shared private information. Here the designer may obscure information (e.g., by not disclosing whether the username or password are incorrect in a login screen) or provide fake information to hide real data (e.g., by adding noise to search results, as is done by statistical databases [1]). Search engines frequently treat users as both clients and adversaries. They provide their users with links to good sites, while protecting against search engine optimizers who want to increase result rank. By hiding and dynamically changing key features of the ranking algorithms, the search engine can serve the need of the user while not falling victim to attacks.

### Individual vs. Self

Perhaps the most ethically fraught type of deception attempts to protect the users from themselves. In these cases, the users' goals (e.g., rehabilitating use of their arms) are in conflict with their behavior (e.g., lack of attempts at more difficult exercises). Likewise, designs can increase safety and help users avoid mistakes, such as by not deleting a file when the user moves it into the trash.

In other situations, an end-user who is not cognitively able to utilize an interface is "handled" through deceptive means via a form of paternalistic deception. A physical-world example illustrates this well. In the town of Düsseldorf, a fake bus stop was set up next to a senior center as a "honey trap" for Alzheimer's patients leaving the facility who would sit at the stop instead of getting lost [18].

### Design Goal vs. Design Goal

Deception can also arise from trying to satisfy design principles. The literature on HCI is filled with rules and sugges-

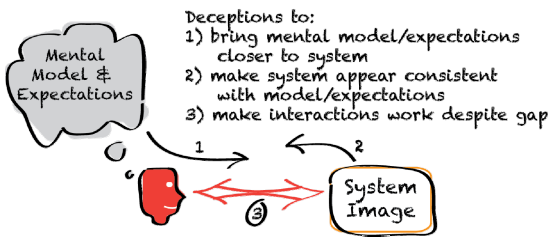


**Figure 3: Means of Deception**

tions about how interfaces should be built. At times, these rules necessitate deception that provides users with better experiences. The principle of least astonishment is occasionally satisfied through the use of deceit. Hiding complexity and providing metaphors may reduce the learning curve and increase adoption of an interface.

In situations in which two design rules conflict, the designer may resort to deception as a way to mitigate the conflict. For example, one should build a system to "fail gracefully" but not to "fail silently." In some cases, however, failing gracefully *is* failing silently and allowing the user to continue. When this happens, it may be appropriate to deceive the user about the existence of the failure.

### Mixed Motive Deceptions

As alluded to earlier, even benevolent deceptions rarely serve the end-user exclusively. The engineer of the deception frequently benefits as well. For example, developers of a product generally make tradeoffs between resources dedicated to building the product and user needs. As money-making enterprises, providing illusions to the user may culminate in real profit to the developer.

The literature on theme park and casino design are filled with examples of this. The theme park lines are especially demonstrative. While increasing enjoyment to individual users (entertaining experiences are provided during the wait, and wait times are overestimated), the lines serve a secondary purpose of allowing the park owner to serve many visitors without requiring costly additions [34, 49].

In casinos, confusing maze-like paths with no easy way out and the illusion of small secluded spaces are created by manipulating architectural elements and encourage users to stay and spend money [20]. Recorded coin noises produced by slot machines are played back to put gamblers into the "zone" (i.e., the machines constantly make noises as if someone has won) [48]. These deceptions are clearly designed to optimize the guest's experience but serve the casino equally by parting the guests from their money.

It is easy to forget that most systems are designed to be used more than once and more importantly, to be sold at a profit (this release, and the next). When running into limited time or other resources there is certainly a temptation to use deception. Both the 1ESS, which does not report failure, and the placebo thermostat, stop complaints and product defections and are at least partially motivated by the benefit to the developer.

### Implications for Research

While we have outlined the various tensions that might motivate deception, in many cases the actual motives are conjectures on our part. Very little research that we are aware of explicitly broaches this topic with designers (what motivates them to build deceptions) or users (their perceptions of motivations and how that affects their experience). A combination of experimental and ethnographic work may serve to clarify these issues.

While many of our examples assume a known "truth," it is worth acknowledging that this is a simplification of a complex epistemological question. In many situations the designer may themselves be unaware of the absolute truth (e.g., a noisy sensor or progress being reported to an end-user). Thus, belief about truth, and truth itself, are highly nuanced concepts that can and should be further explored in understanding deception. It is also important to recognize and study how deception cascades when designers themselves are deceived (e.g., when the designer is the target of benevolent deception).

## THE MEANS

Having discussed reasons benevolent deception might exist in HCI and why researchers and practitioners might choose to use it, we now consider the means by which this deception is enacted. As shown in Figure 3, we categorize three forms by their targets: 1) deception around what the system is doing (system image deceptions); 2) deceptions specifically targeted at the interactions between user and system (behavior deceptions); and 3) deceptions that manipulate the users perceptions of the system (mental model deceptions). However, deception need not, and in many cases cannot, be isolated into only one of these categories.

### System Image Deceptions

As discussed earlier, when the behavior of an underlying system does not match desired objectives, a designer might be motived to manipulate how this information is presented. The deception is in the difference between what the system is actually doing and what is conveyed to the end user. The means by which this occurs can take two forms: deception about what the system is doing (function), and deception about what the system can do (capability).

*Functional deception* frequently reflects incorrect performance information to the user. A common example is the progress bar that smooths, averages, or guesses the actual progress of the system. Functional deception can also mask performance delays, leading users to believe that their actions create an immediate response even when they do not [25]. For example, queues are generally designed to hold data before it is committed to some service. Print, network, mail, or file queues frequently create the illusion of immediate action, when in fact processing is still happening.

Statistical databases [1] also employ this type of deception by only answering queries through aggregate, overly general, or fuzzy responses thus preventing the user from finding the exact data (or even knowing that it exists).

Systems might also be designed to falsely imply the source failure and success. Without preconceptions or specific knowledge, a user is more likely to blame the browser for a failure that occurs while visiting a webpage than the operating system, and to blame the operating system before the computer manufacturer. Deceit about the source of a failure or a success is easy when systems are layered because users typically attribute blame to the level most proximal to themselves. The individual programmer of any component, however, is generally never blamed for anything [43].

In contrast to functional deceptions, *capability-based deceptions* imply something about what the system can do. Often this is done by switching between two systems without telling the user which one is active. The earlier example of how Netflix falls back to a simpler system during periods of high load illustrates this type of deception. Users are unaware of the loss of capabilities when this happens. Similarly, we know of a small vertical search engine designed to search source code that hard-codes the results to certain popular "initial" queries (e.g., "java") rather than perform dynamic searches. This is done because while these queries are a popular initial "test" queries for users unfamiliar with the system, they are not handled well by the system, nor are they representative of the real workload. For this reason, static, manually-crafted result pages, provide high-quality answers, helping users get over the hump of first use.

*Sandboxing* is functional deceit in which a designer creates a secondary system that behaves differently from the real system. For example, the Windows Vista Speech Tutorial is designed to provide the illusion that the user is making use of the real speech recognition system. In reality, a "safe" version of the environment has been created for the user that is programmed to be less sensitive to mistakes [57]. Wizard-of-Oz studies fall into this category by allowing the users to believe they are using one system (a working implementation), but are in fact playing in a human-driven, semi-functional sandbox (though notably such studies frequently include a debriefing that reveals the truth) [16].

While over representing performance and capabilities might be natural, one may wonder about "modest" systems that under-represent their abilities. Such systems emerge to manage user expectations or when safety is an issue. For example, service-based industries (e.g., network, database, server farms, etc.) are fined for not meeting Service Level Agreements. Creating a false impression of available capabilities and consistent performance by throttling back the system is more desirable in that users see constant performance and do not come to expect inconsistently obtainable behaviors. Systems in which safety is a concern may also mislead by using conservative estimates and readings (biased away from the true expected values).

The Time-Sensitive Object Model [10] is an example of a combined performance/sandbox deceit. The system, intended to handle real time data (e.g., from a radar system), had multiple modes of data processing. The first is the presentation of actual real-time data whereas the second extrapolates the value of that data. Similar techniques are applied in modern streaming database applications in which data is dropped if the system becomes overloaded. Thus, the illusion of real-time data processing is maintained by hiding data and performance failures.

System designers might also wish to present a false impression of the *complexity* of a certain interface. The Coinstar company (a manufacturer of coin-counting kiosks found in supermarkets) is rumored to slow down change counting as users were uncomfortable with their money being so quickly processed [6]. In a system built by one of the authors, a security theater was built into the interface. The system was designed to allow users to negotiate for a price using PDAs. The two participants would enter their real asking/offering prices and the system would decide whether a deal was possible (without giving away the prices). Though the particular zero-knowledge protocol was complex and highly secure in the cryptographic sense, it was nonetheless nearly instantaneous and disconcerting for the user (there was no "feeling" that the system was secure). An illusion of complexity was generated by using a slow loop whose only purpose was to replace each entered character with a "*," gradually covering the full text box.

#### Behavioral Deceptions

A user's interaction with a system is mediated by perception, attention, comprehension, prior knowledge, beliefs, and other cognitive activity. From these, a second class of HCI deceits emerge that take advantage of, and occasionally "fix," the *physical*, *sensory*, and *psychological* limits, capabilities, and learned behaviors of the user.

All users have physical and sensory limits that influence their ability to interact. Whether it is in response to limits of perceptions (e.g., color or distance) or resolution (e.g., Fitt's Law), interfaces include deceptive features that attempt to make a user feel more successful. A drop-down menu bar, for example, is programmed not to roll back as soon as the user moves one pixel out of the box. Such practices hide the users' limits from themselves in situations in which the system has perfect sensing but also work well for inaccurate sensors (e.g., certain Nintendo Wii games that give the user "the benefit of the doubt").

Both theme park and interface designers have some understanding of the limits of a user's power of observation (no one can see through a wall.) Thus, while attention to detail is frequently respected to maintain an illusion, things that are not observed by the user are frequently messy or cheap. The developers of the therapy robot, described earlier, took advantage of the perceptual limits of users in a different way. By making use of just-noticeable differences (JNDs), developers can create the illusion that an action has, or has

not, been influenced. The graphics community frequently uses optical illusions, perspective tricks, and cinematographic techniques to force the user to see something that is not there or to ignore something that is. For example, a purposefully blurred image creates the illusion of movement, and only changing a scene when a user's vision is disrupted can make for smoother rendering (i.e., change blindness) [11]. Blindness to change and memory limits may also be used in non-graphic systems, for example, to replace old unused search results with better answers [53].

Systems can be designed to include *showmanship* [54], *eye-candy*, *weenies* [36], *drivers* [20], or *chocolate [41*] to cover mistakes. Regardless of the name, the common feature is that users can be manipulated by distractions into belief or behavior. Image processing systems, because of computational costs and delays, have often made use of this type of deceit. One of the earliest discussions on "placebo" design comes from a graphics system in which a slow-loading image was tiled, slowly appearing on screen, to provide the sensation of action and availability [26].

A wide spectrum of psychological limits, misperceptions, and fallacies can also be used to deceive. Magicians in particular understand such limits, and the application of their ideas in HCI are discussed extensively in [54]. Psychological deceits might also include manipulations based on attractiveness. Aesthetically pleasing designs and faces (e.g., avatars) elicit a more positive response (independent of function) [52]. Psychological deceits based on economic ideas can also be used to motivate behavior (e.g., whether an action is described in terms of risk or gain, the illusion of scarcity, sunk cost fallacy, and relative memories [9]).

Another interesting form of deceit is based on social-psychology, which bridges the gap between behavior and mental model. The tendency to anthropomorphize the computer may lead users to interact with computers as if it they are real people. Leveraging this belief can help system designers build heuristics and features into their system (see sidebar). The Grunt system [46], for example, implies the capability of speech recognition to the user but in fact simply works on non-verbal analysis (e.g., utterance length, pitch, etc.). While the system is quite good at understanding these non-verbal signals, it would be difficult to ask the user to make them. Rather, it is better for the user to more naturally address the system as if it were human.

Similarly, the Phone Slave system [46] made use of "ritualized interactions" by priming the user to talk to an automated agent as if it were a human. By asking leading questions the system forced behavior and consequently did not need to have any real speech recognition. The designer noted that "although we are not trying to deliberately mislead the caller, the illusion of intelligence, either the assumption that one is talking to a human or that one is talking to a very clever machine certainly aids the interaction"

**Mental Model Deceptions**

In many situations designers would like to control the user's Mental Model of the system. In some ways this is the exact opposite of the system image deceptions. Rather than bringing the image more in line with what the user expects, mental model deceptions seek to bring the users expectations more in line with what the system is or can do.

One method for achieving this is through the use of *metaphors*. The designer implies or misleads the user into believing that something works as the metaphor that is describing this item does. Metaphors are rarely acknowledged as deception, but naturally fall into the role by creating the scaffolding that holds the mental model in relation to the system image. Metaphors may not be intentional deception but may nonetheless deceive as, "[they] are basically devices for understanding and have little to do with objective reality, if there is such a thing." [31]

While popular in HCI for their ability to map the unfamiliar to the known, metaphors have been noted by some to "become like a lie…more and more things have to be added to [them]" [38]. Kay instead proposes that where metaphor ends, magic should begin, so that a "user illusion" may be created [28]. This shift potentially replaces one form of deceit—in which we repackage one system as another—with a deceit in which we invent new characteristics that reshape the mental model in other ways (the internal guts of a file system are *not* like a desktop, magic, or otherwise.)

Other designers have embraced the idea of the metaphor and encourage the maintenance of these mappings even though they are a *myth* [45] often by adding skeumorphs. Skype phone calls interject fake static noise, in part because users do not like silence (they think the call dropped). However, the particular choice of noise (the static of a traditional phone systems) maintains an artificial connection between how Skype and traditional phone calls function.

A physical camera shutter being replaced by a digital one or a gas-powered engine replaced with an electric one have led designers to integrate fake sounds (a physical shutter or engine noise). Interestingly, in the transition from "physical" to "digital" systems, what was real now becomes the metaphor. It is here when a metaphor can most obviously be seen as a deception in that users will treat the interface differently because they believe that the new object has the same limitations or capabilities as does the old one. For example, a digital camera designer can use any noise for the shutter (a clown honking its nose), but often play a recording of a physical shutter. Thus, the user—not knowing much about what is happening inside—may believe that the device they are holding is somehow mechanical and will treat it differently.

As noted above, many systems encourage the tendency towards anthropomorphism of the computer. Studies on adding personality and realism to computers encourage this type of deceit (e.g., [15]), and the theme park idea of "illusion of life" is also applied in computer systems. In a par-

ticularly sophisticated example, the designers of CrossTalk break the fourth wall and [29] employed the idea of role and meta-role to, "make the user feel that characters are alive." Because users today believe that all behavior is programmed, they are less likely to believe that the "puppet" they are interacting with is real. The designers instead had the computerized agent tell the end-user that they were actors (the meta-role) playing roles. Thus, while not "believ[ing] in the role being real, [the user was 'tricked'] into believing the meta-role is real" and adding to the authentic experience of the entire performance.

### Implications for Research

The categories we have identified as the means for deception are quite broad. More critically, little has been done to understand them in the frame of deception. Additional exploration and comparison between truthful and deceptive interfaces will likely be of use to practitioners in deciding how best to address system gaps and in the construction of reusable design patterns for benevolent deception. Designing experimental methodology for testing the objective costs/benefits (and subjective impressions) of a truthful interface in relation to its deceptive instantiation, would be highly beneficial to the community.

## THE OPPORTUNITY

Given a motive and means, there are at least two possible opportunities for successful benevolent deceit in HCI: 1) when a user wants to be deceived; and 2) when a user will not be able to tell the truth from the deception. Though these may appear obvious, it is not always easy to find opportunities where deceit will not backfire.

Users sometimes possess "willing suspension of disbelief." In such situations, the user would like—consciously or unconsciously—to experience something beyond the capabilities of the medium (e.g., reality in a virtual reality space, simplicity in a complex space, control in an uncontrolled environment, etc.). The game designers for the popular quiz game You Don't Know Jack have pointed out that the user will spend a few minutes trying to identify the conversational limits of the game's AI but will rapidly become bored of this and again suspend disbelief to enjoy the game [23]. In these instances, the user accepts, and may even demand, deceit over truth. Furthermore, by encouraging a "relationship" between the user and computer, end-users are susceptible to truth bias (perceiving the computer as truthful) [50] thus allowing for further deception.

When a user does not want to be deceived, but the designer/programmer is motivated to deceive, opportunities present themselves in uncertainty. When the user is uncertain about which state the system is in (e.g., cannot build an adequate mental model of the system), or the difference between multiple states, there is an opportunity for deceit. For example, such opportunities exist in cases in which the user can't tell the source of the failure (the 1ESS example) or the impact of their actions (the "placebo" buttons).

However, there is a distinction between "successful" and benevolent deception. While each is necessary for use in HCI settings, neither alone is sufficient. We have pointed out that deceit can be useful to various parties, i.e. companies, developers, etc., but choose to assume in this discussion that designers have an altruistic bent (if sometimes hidden) and the ultimate aim is to make their systems useful to the end-user. In fact, aiming for this tends to benefit all parties, and it is the approach we recommend.

### Getting Caught

Thus far, we have ignored one principle issue with deceit, namely that there is a difference between being deceived and realizing that we have indeed been deceived. Just as in human-human interactions, there is an inevitable cost to being "caught." The programmed price-discrimination by Amazon's or perceived discrimination [30] of Mac users on Orbitz elicited strong reactions by users who felt they were being deceived about the price of items [35]. The Orbitz example is especially interesting, as the deception was actually a form of personalization: Mac users tend to stay in luxury hotels, so these hotels were listed first (intended as benevolent). Unfortunately, luxury hotels are more expensive, and this felt like discrimination to Mac users.

A user who has been trained to click on error boxes to dismiss them may be misled into clicking on an advertisement—to her irritation. A user who has been made confident by a simulator or emulator, basic sandbox type deceits, may find her life threatened when using the real system [27]. Though generally not in the terms of deception, there is a great deal of discourse on trust (e.g., [21]) and credibility (e.g., [22]) in HCI environments. The damage to trust through revelation must be carefully evaluated.

When deceits are not carefully crafted and users get confused, they often have to "go under the hood" (i.e., try to figure out the system model) to proceed with their task. This usually leads to massive failure, as debugging is extremely difficult once the user's view of the system has been intentionally manipulated.

### Not Getting Caught

Inter-human deception has some fundamental differences from computer-to-human deception. Computer interfaces, APIs and GUIs, encourage abstraction barriers around complex internal machinery. Unlike communication between two humans, users are less likely to be able to place themselves in the "mindset" of the system as they would with another human and are therefore less likely to detect deceit. Additionally, interfaces tend to be designed for consistency, foiling one of the primary mechanisms by which humans detect deceit among themselves. On the other hand, deceit requires careful adaptation and planning—something programs are generally not very good at (e.g., when a metaphor breaks in some unexpected way). When considering implementing a deceit, it is worth building with these differences in mind. It is also important to consider the impact

of not being caught in the long run. A user may come to rely on a deceit and become unwilling or unable to adapt to new and better interfaces. Once created, many deceits require commitment, forcing a quick "hack" to become a permanent solution.

### Implications for Research

While various results on trust and psychology (and physiology) exist (the literature on magic is highly informative), the lack of integration into the deception frame makes it difficult to understand whether a deception is detectable or desirable. While we have many metrics for determining efficiency, preference, and accuracy, no metric will tell us "how much a user wants to be deceived" or as a measure of "how much the user can be deceived." In addition, deception in HCI may function very differently in different cultures. There is likely a difference in the tolerance, acceptability, and detectability of deception in different countries.

## CONSTRUCTION OF DECEPTION AND ETHICS

Understanding the different means, motives, and opportunities can be used as ingredients for designed deceit (see Figure 4). However, as we note above, there is a difference between blind application and thoughtful design of benevolent deceptions. Ideally, deceit would be considered early in the design process and in the context of all stakeholders, rather than as an attempt to patch a mistake. Additional research is vital here to inform practitioners of nuances of designed deception. When a benevolent deception is well-crafted and encompasses desired critical functionality, users typically benefit. However, when users need to look behind the deception to perform their tasks, results are sometimes catastrophic, as users are not usually equipped to debug the system that has been intentionally hidden from view.

### Ethical Considerations

While we have addressed some of the ethical issues around deception, it is worth returning to this once again. An important step is understanding our motivations, as designers or researchers, for creating the deceit. Though we have listed potential motives, it is important to acknowledge that many are simply rationalizations that require careful analysis. We clearly do not endorse dogmatic views of deceit (always good or always bad). However, as argued by Bok [5] in the case of lying, we believe that given the negative impacts of deceit it is worth considering all possible truthful alternatives (a step 0). Again, this is an area ripe for research and exploration. While the ethics of deception of certainly have been studied in many other disciplines (philosophy, law, etc.) they are relatively untouched in HCI.

## CONCLUSIONS AND FUTURE RESEARCH

In this paper we have we have provided an alternative view of system and interface design and the use of benevolent deception to influence user behavior. From the perspective of practice, while we obviously do not advocate blindly resorting to deceit in all instances, we believe that a) designers do this all the time without realizing it, and b) there are opportunities in benevolent deception for improving our understanding and ability to craft useful systems. Deceit is a frequently used, but rarely designed, feature in HCI. We attempt to provide preliminary guidance on design principles that emerge from the descriptive model we present, but as with any good set of principles, we believe this requires deeper discussion and continued iteration within the community. The purpose of this paper is to propose a backbone for such discussion, shed light on the issues, and provide a starting point for such discussion.

Construction of new deceptions requires working backwards through the definition of the deceit by

1. selecting who we want to deceive,
2. deciding what behavior or belief we intended to influence,
3. understanding how a user will process the deceit,
4. selecting from the various types of deceits that work in HCI contexts the appropriate means that will produce the behavior, and
5. carefully seeking the opportunity to commit the deception

**Figure 4: How to design for benevolent deception.**

From the perspective of research, we hope this paper inspires both discussion and further research of benevolent deception. At the very least we believe that it is vital to stop treating deception as taboo. By failing to discuss deception in those terms, it becomes difficult to fully understand the impact on the users and their experiences of the system. We have outlined a set of research questions throughout the paper that we believe are worthy of pursuit: understanding motives for deception; the user's perception of motive; patterns of deception; measures of effectiveness of deception; and cultural and ethical concerns.

Armed with this information, both designers and researchers alike will, we hope, be inspired to understand and to use deceptions for good committing these "crimes of deception" *for* rather than *against* users.


## ACKNOWLEDGEMENTS

We're particularly grateful to those who read early versions of this paper: Mark Ackerman, Sara Adar, Yaw Anokwa, Michael Bernstein, Batya Friedman, Travis Kriplean, Yang Li, and Dan Weld. We would also like to thank our many friends and colleagues who pointed out great cases studies.


## REFERENCES


1. Adam, N.R., and J.C. Worthmann, "Security-control methods for statistical databases: a comparative study," *ACM Computing Surveys*, 21(4):515-556, Dec. 1989.
2. Arabe, K. C., "'Dummy' thermostats cool down tempers, not temperatures," *Industry Market Trends*, Apr. 11, 2003.
3. Armitage, T., "Conversational UI: a short reading list," bit.ly/dFG77o, Dec. 10, 2010.
4. Bell, J. B., and B. Whaley, *Cheating and Deception*, Transaction Publishers, 1991.

5. Bok, S., *Lying: Moral Choice in Public and Private Life*, Vintage Press, 1999.
6. Brignull, H., "Adding delays to increase perceived value: does it work?" bit.ly/i7lSCf, Dec. 16, 2010.
7. Boehner, K. and J. T. Hancock, "Advancing ambiguity," CHI 2006.
8. Brewer, B.R., M. Fagan, R. Kaltzky, and Y. Matsuoka, "Perceptual limits for a robotic rehabilitation environment using visual feedback distortion," *IEEE Trans. on Neural System and Rehabilitation Eng.*, 13:1-11, 2005.
9. Camerer, C. F., G. Loewenstein, and M. Rabin (eds.) *Advances in Behavioral Economics*, Princeton Press, 2003.
10. Callison, H. R., "A time-sensitive object model for real-time systems," *ACM TOSEM*, 4(3):287-317, July, 1995.
11. Carter, K., A. Chalmsers, and C. Dalton, "Level of detail: varying rendering fidelity by exploiting human change blindness," Int. Conf on Comp Graphics and Int. Techniques in Aust. and South East Asia, 2003.
12. Ciancutti, J., "5 lessons we've learned using AWS," Dec. 16, 2010, Available at: nflx.it/hrjm6J.
13. Cockton, G., "Value-centered HCI," NordiCHI, 2004.
14. Conti, G., and E. Sobiesk, "Malicious interface design: exploiting the user," WWW, 2010.
15. Cowell, A. J., and K. M. Stanney, "Manipulation of non-verbal interaction style and demographic embodiment to increase anthropomorphic computer character credibility," *Int. J. of Human-Computer Studies*, 62:281-306, 2005.
16. Dahlbäck, N., A. Jönsson, and L. Ahrenberg "Wizard of Oz studies: why and how," IUI, 1993.
17. Dark Patterns Wiki, http://wiki.darkpatterns.org.
18. de Quetteville, H., "Fake bus stop keeps Alzheimer's patients from wandering off," *Telegraph*, Jun. 3, 2008.
19. Erat, S., and U. Gneezy, "White lies," *Management Science*, Article in Advance, online Nov. 4, 2011.
20. Friedman, B., *Designing Casinos to Dominate the Competition*, ISGCG, University of Nevada, 2000.
21. Friedman, B., *Human Values and the Design of Computer Technology*, CSLI, 1997.
22. Fogg, B. J., *Persuasive Technology: Using Computes to Change What We Think and Do*, Morgan Kaufmann, 2002.
23. Gottleieb, H., "The Jack Principles," Jellyvision Inc., bit.ly/9pYnaM, 1997.
24. Hancock, J.T., J. Thom-Santelli, and T. Ritchie, "Deception and design: the impact of communication technology on lying behavior," CHI, 2004.
25. Harrison, C., Yeo, Z., and Hudson, S. E., "Faster progress bars: manipulating perceived duration with visual augmentations," CHI, 2010.
26. Irani, K.B, V. L. Wallace, and J.H. Jackson, "Conversational design of stochastic service systems from a graphical terminal," Int. Symp on Comp Graphics, 1970.
27. Johnson C.W., "Looking beyond the cockpit: human computer interaction in the causal complexes of aviation accidents," HCI in Aerospace, EURISCO, 2004.
28. Kay, A., "User interface: a personal view," In B. Laurel (ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley, 1991.
29. Klesen, M., M. Kipp, P. Gebhard, "Staging exhibitions: methods and tools for modeling narrative structure to produce interactive performance with virtual actors," *Virtual Reality*, 2003, 7(1):17-29.
30. Krugman, P., "What price fairness?," *The New York Times*, Oct. 4, 2000.
31. Lakoff, G., and M. Johnson, *Metaphors We Live By*, University of Chicago Press, 1980.
32. Laurel, B., *Computers as Theatre*, Addison-Wesley, 1993.
33. Luo, M., "For exercise in New York futility, push button," *The New York Times*, Feb. 27, 2004.
34. Marling, K.A. (ed.), *Designing Disney's Theme Parks: The Architecture of Reassurance*, Flammarion, 1997.
35. Mattioli, D., "On Orbitz Mac users steered to pricier hotels," *Wall Street Journal*, August 23, 2012.
36. Mine, M., "Towards virtual reality for the masses: 10 years of research at Disney's VR Studio," Eurographics, 2003.
37. Mitrašinovi, M., *Total Landscape, Theme Parks, Public Space*, Ashgate, 2006.
38. Nelson, T. H., "The right way to think about software design," In B. Laurel (ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley 1991.
39. Norman, Donald, "Some observations of mental models," In Gentner, Dedre & Stevens, Albert L. (eds.), *Mental Models*, Lawrence Erlbaum Associates, 1983.
40. OCLC, "Fourteen heuristics used in OCLC heuristic evaluations," http://www.oclc.org/, retrieved Jan. 2008.
41. Plauger, P. J., "Chocolate," *Embedded Systems Programming*, 7(3):81-84, Mar. 1994.
42. Rams, D., "Ten principles for good design," bit.ly/QUk4gY, retrieved Sept. 2012.
43. Reeves, B., and C. Nass, *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*, CSLI, 2003.
44. Richards, J.I., *Deceptive Advertising: Behavioral Study of a Legal Concept*, Lawrence Erlbaum Associates, 1990.
45. Rubinstein, R., and H. M. Hersh, *The Human Factory: Designing Computer Systems for People*, Digital, 1984.
46. Schmandt, C., "Illusion in the interface," in *The Art of Human-Computer Interface Design*, Addison-Wesley, 1991.
47. Sandberg, J., "Some cold employees may never find relief," *Wall Street Journal Online*, Jan. 17, 2003.
48. Schull, N. D., "Digital gambling: the coincidence of desire and design," *Annals of the Am. Ac. Of Poli. And Soc. Sci.*, January 2005, 597(1):65-81.
49. Stone, A., "Why waiting is torture," *New York Times*, Aug. 18, 2012.
50. Stuff, J.B., H. J. Kim, and C.N. Ramesh, "Truth biases and aroused suspicion in relational deception," *Communications Research*, 1992, 19:326-345.
51. Takayama, L., D. Dooley, W. Ju, "Expressing thought: improving robot readability with animation principles," HRI, 2011.
52. Tractinsky, N., A.S. Katz, and D. Ikar, "What is beautiful is usable," *Interacting with Computers*, 13:127-145, 2000.
53. Teevan, J., "The Re:Search Engine: simultaneous support for finding and re-finding." UIST, 2007.
54. Tognazzini, B., "Principles, techniques, and ethics of stage magic and their application to human interface design," INTERCHI, 1993.
55. West, Mick, "Intelligent mistakes: how to incorporate stupidity into your AI code," bit.ly/Nq0sCL, September 15, 2012.
56. Weyenberg, A., "Is realistic UI design realistic?" bit.ly/9b5Tr0, retrieved Oct. 2010.
57. Zheng, L, "Interview with Oliver Scholz: Vista Speech UX program manager," bit.ly/dCIRWH, Sep. 1, 2006.