Using Deceit in HCI: Crimes For or Against the User?

Eytan Adar
University of Washington
101 Paul G. Allen Center
Seattle, WA 98195-2350
eadar@cs.washington.edu

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

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

CHI 2008, April 5 – April 10, 2008, Florence, Italy
ACM 1-xxxxxxxxxxxxxxxxx.

Abstract
The use of deceit in human-computer interaction is generally rejected by designers and ignored in HCI research, despite the fact that well-used deception can have a significant positive impact on user experience. In this paper we present a model of deceit that builds upon a criminal metaphor. With numerous examples, we explore how designers may find the means, motive, and opportunity to commit "crimes" of deception, and discuss the benefits and costs to systems and users.

Keywords
Deceit, deception, lying, crime, means, motive, opportunity.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

Introduction
Much human-computer interaction (HCI) research has centered on increasing the communication bandwidth between humans and computers so that users can complete their tasks more efficiently. Researchers have traditionally aimed to create systems and interfaces that enable users to directly perform their desired actions. In this paper, we present an alternate view, framing HCI work within a paradigm focused on the controlled use of deceit to manipulate user actions so

that users better achieve their goals. In doing this, we borrow from terminology used in criminology. Through numerous examples, we show that this paradigm can be used to retroactively describe previous research, and hope that it will motivate discussion as well as novel ways of thinking about and working in the space.

Due to negative associations, deceit is generally rejected by designers and ignored in research. The impression is typically that outright deception should not (and does not) exist in good design, and that misleading information is simply the result of a bug or poor design (e.g., "printing successful," *lied* the computer with the failed print driver). However, there are many examples in which deception and deceptiveness may in fact represent positive design choices. The reality is that whether intentional or unintentional, implicit or explicit, acknowledged or not, deceit exists in HCI. We assert that with proper understanding, we can embrace deceit for the benefit of users and/or designers. Take the following case studies as examples:

Example 1: 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 1ESS, the first such system, was designed to provide reliable phone communication, but given the restrictions of early 1960s hardware, sometimes had unavoidable failures that could lead to errors in initial connection. However, the 1ESS was designed to not report failures to the client (though it could). Instead, erroneous calls were allowed to go through, connecting the caller to the wrong person. The individual, thinking that they had simply misdialed, would hang up and try again, and the illusion of the infallibility of the phone system was preserved [24].

Example 2: In order to help stroke patients regain movement, researchers designed a custom robot to provide Constraint-Induced Movement Therapy. The robot, which was attached to a monitor, provided visual feedback to the user on the amount of force exerted. In order to "overcome…self-imposed limits," the system was designed to leverage perceptual limits through visual distortion. For example, a patient may not be able to distinguish between 7 and 9 units of effort, but may believe themselves only capable of 8 units. The system would visually report 7 when the patient had exerted 8, inducing the feeling that they had exerted less effort and encouraging additional force [4].

Example 3: People generally like to feel like they control their environment. When a system designer, for whatever reason, does not provide actual control they may introduce "placebo buttons." For example, in 2004, 2500 of 3250 cross-walk buttons in New York City were implemented even though they did not function at all [18]. Similarly, buttons to control the elevator or the thermostat may provide the illusion of control but may in fact be a buttons connected to nothing [29]. Though we may partly justify such designs based on user needs and desires, in reality this design decision may serve the designers, developers, or system owners more.

A common feature of the examples, and HCI deceit in general, is the manipulation of a user's belief (i.e., mental model) relative to the true properties of the system (i.e., the system image or system model). The difference between reality and belief, and the frequent poor fit between them, is the region in which deceit, deception, and deceptiveness may exist.

In this paper, we present a model of deception based on an analogy to criminology, couching it in terms of

means, *motive,* and *opportunity*. This model allows us to look at how previous systems have taken advantage of the poor fit between a user's mental model and reality. We conclude with a discussion of how and when such a model might be used to create an experience for the user that is better than might be achieved without the use of deception.

Related Metaphors

The closest work in this area specific to HCI has been the use of magic [32] and cinema/theater [16] as instructional metaphors for HCI design. These art-forms—where illusion, immersion, and the drive to shape reality dominate—work well in situations in which there is willing suspension of disbelief on the part of the user/observer (or at least willing pursuit of acceptable mental models). Similar lessons may be drawn from architectural design. For example, theme park and casinos [9][19][20] are designed specifically to utilize various illusions that manipulate users' perception of reality, entertainment, and participation in the experience. In the case studies below we will see a number of deceits in HCI systems that have parallel designs to magic, theater, and architecture.

In the rest of this paper, we expand previous metaphors to include instances in which the user does not willing participate in the deception. Though the notion of HCI deceit as a crime is somewhat sensationalist, it works well as deception and deceptiveness are traditionally regarded as violations of design rules, principles, and laws (e.g. [3][10][23][27]), albeit often for good cause.

Note that we are not concerned with deceits that are violations of criminal law (e.g., phishing and other fraud). Deceptive practices that are acknowledged as harmful by legal organizations are, for generally good reasons, considered harmful by designers.

Deceit in the language of criminology further allows us to begin to understand the "victims," i.e., the users, who would act differently had they known the truth. This language opens deceit to analysis in terms of *means*, *motive*, and *opportunity* (*MMO*), three well-known aspects used to analyze crimes. However, before concentrating on the who, why, when, where and how of deceit, it is important to form some formal understanding of what deceit is.

A Working Definition of 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, the FCC will only consider the deceptiveness of the message irrespective of intent. That being said, proving motive/intent in a system or design is also a very convincing argument for conviction. There is a notable difference between un-intentional bugs, errors, or bad metaphors and ones that have been carefully designed for specific purpose.

Building on the behavioral/legal definition introduced in earlier work [26] that deals with deceptive advertising, we put forth a working definition of deceit as it applies to HCI work in the sidebar. The points about how deceit

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.*

(Definition based on [26])

must substantially affect behavior (points 5 & 6) are perhaps the most controversial, and are purposefully left ambiguous. How behavior is impacted and what "substantial" means are left open since there is likely no fixed 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 ways of dividing deceit into sub-categories. Bell and Whaley [2] 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 verbal or expressed deception. *Lying*, as a special class, is generally considered to be a verbal form of deception [3]. Because HCI need not involve a verbal element, we expand the notion of the "lie" to include non-verbal communication between humans and computers.

The Means

As mentioned to earlier, deceit exists in the area between the reality of the system and the mental model. A deceptive or misleading system works by manipulating or taking advantage of the distance between the system image and that mental model. We propose a number of ways in which this happens in HCI work. Note that categories in our taxonomy are not orthogonal, and specific case studies frequently involve multiple forms of deceit.

Functional Deceits

Functional deceit works by misleading the user about how something works. In certain cases we see instances of the user's mental model being adjusted through deceit. In other examples, the system image will be changed or deceptively given the appearance of change, to more closely match the mental model and user expectations.

One of the most common forms of functional deceit is the use of *metaphors*. The designer implies or misleads the user into believing that something works as the metaphor by which it is being described. Metaphors are rarely acknowledged as deception, but naturally fall into the role by creating the scaffolding that holds the mental model separate from the 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." [16]

While popular in HCI for their ability to map the unfamiliar to the known, some (e.g., [21]) have noted that metaphors "become like a lie…more and more things have to be added to [them]." Kay instead proposes that where metaphor ends, magic should begin, so that a "user illusion" may be created [13]. 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 but are nonetheless different than reality (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* [27]. In

addition to the popular desktop metaphor, many encourage the tendency towards anthropomorphism of the computer. Studies on adding personality and realism to computers encourage this type of deceit (e.g., [8][16]), and the theme park idea of "illusion of life" is also applied in computer systems.

Informational Deceits

In *informational* deceit, users are led to believe that their action created an immediate response that typically masks system delays. 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] are a different example of such programmed informational deceits. These databases are designed to only answer queries through aggregate, overly general, or fuzzy responses that prevent the user from finding the exact data (or even knowing that it exists).

Systems may be designed to include *showmanship* [32], *eye-candy*, *weenies* [19], *drivers* [9], or *chocolate* [24] 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 [12].

Sandboxing

Sandboxing is functional deceit in which a designer creates a secondary system that behaves differently from the real system. For example, the 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 [33]. Wizard-of-oz studies fall into this category by allowing the user to believe that they are using one system (a working, programmed implementation), but are in fact playing in a human-driven, semi-functional sandbox.

System Internals: Performance, Complexity, and Failure

Frequently functional deceits and sandboxing can be used for the purpose of implying certain *performance* levels. For example, we know of at least one search engine (a small vertical one) that has responses to certain popular queries hard-coded. Realizing that new users frequently try the system with these popular queries, the system provides high-quality answers to them. Performance based deceits can work in both directions, misleading users into believing the system is under- or over- performing or about the *capabilities* of the system. The Grunt system [28] 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 over-representing performance and capabilities might be a natural, one may wonder about "modest" systems that under-represent their abilities. Such systems emerge in situations where user expectations need to be managed or safety is an issue. For example, service-based industries (e.g., network, database, server farms, etc.) are fined heavily for not meeting Service Level Agreements (SLAs). Creating a false-impression of available capabilities and consistent

Masked Complexity

One example is a system one of us designed to allow users to negotiate for a price using PDAs. The two participants would enter their real asking/offering prices into their PDAs 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 from the perspective of 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.*

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 make use of conservative estimates and readings (biased from the true expectation and levels) that effectively mislead a user.

The Time-Sensitive Object Model [6] 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 where 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 may also wish to present a false impression of the *complexity* of a certain interface (see sidebar). A related idea is the multi-level interface. An expert user may be presented (or have some back-door access) to one form of the interface, giving them access to many functions and controls. The novice, who is provided with a much simpler view, is led to believe that the system is less complex than it is.

Finally, systems may be designed to falsely imply the source of *failure/success*. The ESS example discussed earlier is a clear example of such a deception, as the user is led to believe that the failure (misdialing) was their fault. Deceptions such as these are easy where there are layered systems since users will frequently attribute blame to the level most proximal to themselves. In the absence of other factors such as preconceptions or knowledge of failure sources, a user is more likely to blame the browser than the operating system and the operating system would be blamed before the computer manufacturer. It is interesting that the individual programmer of any component is generally never blamed for anything [25].

Human Factor Deceits

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

All users have some physical/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 user's limits from themselves in situations where 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

Ritualized Interactions

The Phone Slave system made use of "ritualized interactions" by priming the user to talk to an automated agent as if it were a real person. By asking leading questions that made users respond the way they would to a human, the system 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" [28].

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 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) [7]. 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 [31].

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 [32]. Psychological deceits may also include manipulations based on attractiveness. Aesthetically pleasing designs and faces (e.g., avatars) are known to elicit a more positive response (independent of function) [30]. 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 [5]).

Another interesting form of deceit is based on social-psychology. As mentioned earlier, 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 Motive

In our definition of deceit we noted that depending on intent we either have deception or deceptiveness. Understanding the difference helps us to decide if the design is deceptive and contains unintentional deceptive elements, such as bad metaphors, unclear user models, and interface bugs. Alternatively, the design may be intentionally deceptive for various reasons. Though this deceit need not be malicious or malevolent, the designer or programmer has made a conscious decision to deceive the user.

Thus far, we have concentrated on the user's mental model and the system image in understanding deceit. A third part of this relationship is the designer's mental model (i.e. the design) which represents the goal of the designer. Abstractly, if either the system image or user's mental model are not aligned with the design, the designer/developer may resort to deception. This appears especially true when designs must create balance between maximizing the utility to the user, minimizing "risks," or maximizing the "profit" of the designer or developer.

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 such examples. The theme park must manage long lines and use demands that exceed the resources of the park. Deceptive lines and other distractions allow users of the park to enjoy their visit without requiring more costly additions [19]. 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 [9].

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 (hopefully buying the next version as well). When running into limits of time or other resources there is certainly a temptation to use deception to satisfy the user. Both the 1ESS which doesn't report failure, and the placebo thermostat, stop complaints and product defections and are at least partially motivated by the benefit to the developer.

Deceits that attempt to help the user are generally easier to excuse (unless you are a believer in psychological egoism). The literature on HCI is filled with rules and suggestions about how interfaces should be built. At times, these rules necessitate certain deceits that provide users with better experiences. For example, 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. Other deceptions are motivated (and justified) by the fact that they will help the user later on. The therapeutic robot is one such example.

In situations where two design rules conflict, the designer may also resort to deception as the "lesser of two evils." For example, one should build a system to "fail gracefully" but not to "fail silently." But what of the situation in which failing gracefully *is* failing silently and allowing the user to continue? In such cases, the user may be deceived about the existence of the failure.

A different driver for deceit is that some systems must serve the needs of many users and the interactions between them. The need to maintain security and privacy might lead to designs like the statistical databases described above or the login screen that does not disclose which of the two fields, username or password, were incorrect. Situations in which we have an "adversary" (e.g., malicious users, in military applications, etc.) are frequent candidates for deceptive behaviors (e.g., honeypot servers that appear as regular unprotected machines but are meant to trap hackers or network providers that deceive file sharing applications into believing they are disconnected).

Although we are primarily concerned with HCI in this paper, it is worth noting that computer-mediated communication systems are occasionally designed to be manipulated by one user (the "programmer" of the message) to convey false information as in an instant messaging system that provides the ability to hide presence, or e-mail systems that provide the illusion of availability through delayed mail and automatic responses.

The Opportunity

These are at least two possible opportunities for (successful) deceit: a) when a user wants to be deceived, and b) 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.). In these instances, the user accepts, and may even demand, deceit over truth.

Designed Deception:

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*

When a user does not want to be deceived, but the designer/programmer is motivated to, 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 where the user can't tell the source of the failure (the 1ESS example) or the impact of their actions (the "placebo" buttons).

However, it should be noted that there is a distinction between "successful" and "useful" deception and while each is necessary for use in HCI settings, neither alone is sufficient. While we have pointed out that deceit can be useful to various parties, i.e. companies, developers, etc., we 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.

Implications, Consequences, and Ethics

Having considered various case studies of deceit in the context of HCI, we now turn to the broader implications of our analysis and deceit in general. Understanding the different means, motives, and opportunities can be used as ingredients for designed deceit (see sidebar). However, as we note above, there is a difference between blind application of the recipe and more thoughtful design of useful deceptions. Ideally, deceit would be considered early in the design process, and in the context of all stakeholders, rather than as an attempt to quickly patch a mistake.

Useful Deception: There are many opportunities for useful deceit, most of which revolve around intentionally creating a user mental model that does not correspond to what the system is actually doing. In situations where a system that is deficient or limited in some way—too complex, too slow, too uncertain—and that deficiency interferes with the intended purpose of the system, a designer may choose to hide these negative properties. Through deception, the designer may provide the appearance that the negative properties do not exist. Though subtly different, the inverse of this is when the user or designer expects the system to display positive properties that are not present. Here, the design must create the illusion that certain features are available.

Both manipulations involve no real change to the core system image but rather "tweak" the user's model of that image. Regardless of why or how, when this model is well-crafted and encompasses desired critical functionality, users typically benefit. However, when users have to sidestep this model to perform their tasks, results are sometimes catastrophic as users are not usually equipped to debug the system which has been intentionally hidden from view.

Getting Caught: Thus far we have ignored one principle issue with deceit, namely that there is a difference in being deceived, and realizing that we have been deceived. Just as in human-human interactions, there is an inevitable cost to being "caught." The programmed price-discrimination on Amazon's website elicited strong reactions by users who felt they were being deceived about the price of items [15]. A user that has been trained to click on error boxes to dismiss them may be misled into clicking on an advertisement to their irritation. A user that has been made confident by a simulator or emulator, basic sandbox type deceits, may find their life threatened when using the real

system [13]. Though generally not in the terms of deception, there is a great deal of discourse on trust (e.g., [10]) and credibility (e.g., [11]) in HCI environments. The damage to trust and the user on revelation of a deceit 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) in order 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: Human-to-human deception has some fundamental differences to computer-to-human deception. Computer interfaces, APIs and GUIs, tend to 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 between themselves. On the other hand, deceit sometimes 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 a deceit, it is worth building with these differences in mind.

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

Ethical Considerations: An important step in designed deception is understanding our motivations for creating the deceit. Though we have listed potential motives, it is important to acknowledge that many are simply rationalizations which require careful analysis. We do not endorse dogmatic views of deceit (always good or always bad). As argued by Bok [3] in the case of lying, we believe that given the negative impacts of deceit it is worth considering all possible truthful alternatives. It is therefore worth adding a step 0 to our designed deception: 0) exhausting all truthful options.

Conclusions

In this paper we have we have provided an alternative view of system and interface design that borrows from the lexicon of criminology and couches much of our work within the careful use of deceit in order to influence user behavior. While we obviously do not advocate blindly resorting to deceit in all instances, we believe that there are opportunities in the metaphor 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 fall out of 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.

We assert that the end-goal and motivation for using deceit must be carefully considered. We hope that this article inspires designers to understand and use these powerful tools for good, committing these "crimes of deception" for rather than against users. Similarly, armed with this information, we hope that the criminals

become easier to apprehend, and that the CHI community as a whole will benefit from this alternate view of the work we do.

Acknowledgements

We're particularly grateful to those who read early versions of this paper: Dan Weld, Yaw Anokwa, Travis Kriplean, Sara Adar, and Yang Li. Thanks also to Yoshi Kohno, David Notkin, Brian Bershad, Andy Begel, and Mike Toomim for suggested case studies. Eytan Adar is funded by an NSF and an ARCS Fellowship.

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] Bell, J. B., and B. Whaley, *Cheating and Deception*, Transaction Publishers, 1991.

[3] Bok, S., *Lying: Moral Choice in Public and Private Life*, Vintage Press, 1999.

[4] 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 Engineering*, 13:1-11, 2005.

[5] Camerer, C. F., G. Loewenstein, and M. Rabin (eds.) *Advances in Behavioral Economics*, Princeton Press, 2003.

[6] Callison, H. R., "A time-sensitive object model for real-time systems," *ACM Trans. on Software Engineering and Methodology*, 4(3):287-317, July, 1995.

[7] Carter, K., A. Chalmsers, and C. Dalton. (2003). Level of detail: Varying rendering fidelity by exploiting human change blindness. Int. Conf on Comp Graphics and Interarctive Techniques in Aust. and S. East Asia, 2003.

[8] 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.

[9] Friedman, B., *Designing Casinos to Dominate the Competition*, ISGCG, University of Nevada, 2000.

[10] Friedman, B., *Human Values and the Design of Computer Technology*, CSLI, 1997.

[11] Fogg, B.J., *Persuasive Technology: Using Computes to Change What We Think and Do*, Morgan Kaufmann, 2002.

[12] 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.

[13] Johnson, C.W. "Looking beyond the cockpit: human computer interaction in the causal complexes of aviation accidents" HCI in Aerospace, EURISCO, 2004.

[14] Kay, A. "User Interface: A Personal View," In B. Laurel (ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley, 1991.

[15] Krugman, P., What Price Fairness?, *The New York Times*, October, 4, 2000.

[16] Lakoff, G., and M. Johnson, *Metaphors We Live By*, University of Chicago Press, 1980.

[17] Laurel, B., *Computers as Theatre*, Addison-Wesley, 1993.

[18] Luo, M., "For Exercise in New York Futility, Push Button," *The New York Times*, Feb. 27, 2004.

[19] Marling, K.A. (ed), *Designing Disney's Theme Parks: The Architecture of Reassurance*, Flammarion, 1997.

[20] Mitrašinović, M., *Total Landscape, Theme Parks, Public Space*, Ashgate, 2006.

[21] 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.

[22] Norman, D. "Some Observations of Mental Models," In Gentner, Dedre & Stevens, Albert L. (eds.), *Mental Models*, Lawrence Erlbaum Associates, 1983.

[23] OCLC, Fourteen heuristics used in OCLC heuristic evaluations, http://www.oclc.org/policies/usability/heuristic/set.htm, retrieved Jan. 2008.

[24] Plauger, P. J. "Chocolate," *Embedded Systems Programming*, 7(3):81-84, Mar. 1994.

[25] Reeves, B., and C. Nass, *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*, CSLI, 2003.

[26] Richards, J.I., *Deceptive Advertising: Behavioral Study of a Legal Concept*, Lawrence Erlbaum Associates, 1990.

[27] Rubinstein, R., and H. M. Hersh, *The Human Factory: Designing Computer Systems for People*, Digital, 1984.

[28] Schmandt, C., Illusion in the Interface, In B. Laurel (ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley, 1991.

[29] Sandberg, J., "Some Cold Employees May Never Find Relief," *Wall Street Journal Online*, Jan. 17, 2003.

[30] Tractinsky, N., A.S. Katz, and D. Ikar, "What is beautiful is usable," *Interacting with Computers*, 13:127-145, 2000.

[31] Teevan, J. "The Re:Search Engine: Simultaneous support for finding and re-finding," UIST, 2007.

[32] Tognazzini, B., "Principles, Techniques, and Ethics of Stage Magic and Their Application to Human Interface Design," INTERCHI, 1993.

[33] Zheng, L. "Interview with Oliver Scholz: Vista Speech UX Program Manager," www.istartedsomething.com/ 20060901/interview-oliver-scholz/, Sep. 1, 2006, (retrieved Jan. 2008).