Direct Manipulation

Sebastian Fichtner
sebastian.fichtner@uni-konstanz.de

ABSTRACT

To many interaction designers, the term *direct manipulation* seems either too fuzzy or too plain obvious. In both cases, it isn't very helpful. In this seminar paper, instead of reiterating *dos* and *don'ts*, we tell the story of direct manipulation and dive down into its theoretic foundation. By integrating different academic perspectives, we clarify related concepts and make the direct manipulation philosophy accessible as a pragmatic way of thinking about interaction. Thereby, we propose an accommodated interface space and show that direct manipulation is still at the core of most of today's innovations in human-computer interaction.

1. PARADIGM CHANGE

When the Xerox Star machine was introduced in 1981, we entered the era of windows, icons, menus and pointers.

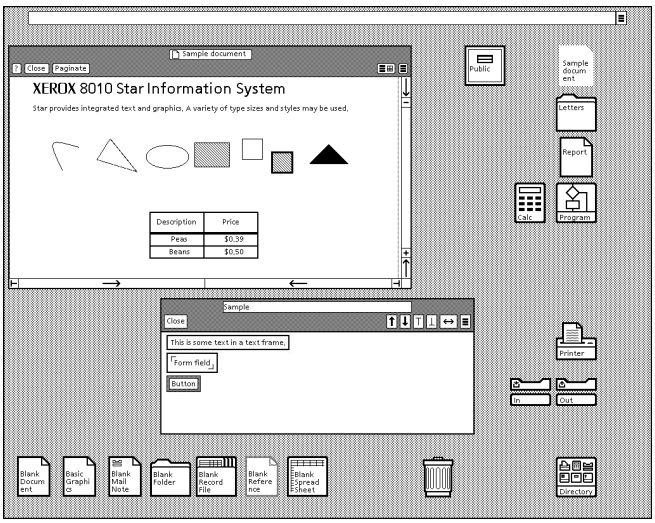

Figure 1. The XEROX 8010 Star Information System

Figure (1) depicts the Xerox Star interface. Steve Jobs copied that interaction concept and implemented it in the Apple Lisa, which became the first commercial computer to offer a graphical interface. For more than 30 years, WIMP has dominated the interfaces of operating systems. It's what we used everyday. In fact, we've been exposed to the WIMP interaction style for so long that we've come to confuse it with its idea. Today, as we're moving on to mobile operating systems, the WIMP era is coming to an end, but the story of its underlying design philosophy is just getting exciting. To understand that story, we have to start at its beginning.

Even before the rise of computer graphics, when interfaces were still character based, a new quality of interaction emerged. The most prominent example is VisiCalc. It was the first spreadsheet software ever and mysteriously successful (Figure 2).

Figure 2. The VisiCalc interface

Today, we identify four key features that made it special:

- The table it displays was something very common in the business world. Everyone understands how to use a table- and what to use it for.
- All numbers are visible at once.
- All numbers can be changed.
- The effects of those changes on subsequent calculations are immediately visible.

From our modern perspective, those features might not seem so groundbreaking, but VisiCalc marks a paradigm change. When it was released in 1979, it became the killer application for the Apple II.

The first academic to point out the pattern behind such successful applications was Ben Shneiderman. In 1982, one year before Apple released the Apple Lisa, he coined the

Theories & Models in HCI, Prof. Harald Reiterer, Hans-Christian Jetter
Konstanz University, 2012/13

term *direct manipulation* (DM) for the principles he had observed [14]. He described a certain outstanding quality of user satisfaction along with the kind of interface that seemed to cause it. According to Shneideman, users reported the following experiences:

- Mastery of the system
- Competence in performance of their task
- Ease in learning the system originally- and in acquiring advanced features
- Confidence in their capacity to retain mastery over time
- Enjoyment in using the system
- Eagerness to show it off to novices
- Desire to explore more powerful aspects of the system

This informal observation may seem a little unscholarly, but we have to acknowledge that, at the time, the sensitivity to human factors of software was just developing. Shneiderman stated in the same '82 publication:

> "The egocentric style of the past must yield to humility and a genuine desire to accommodate to the user's skills, wishes and orientation."

To Shneiderman, DM wasn't just an observation. It had to be the first consequence of embracing the user.

Other authors had already alluded to single aspects but didn't grasp the whole pattern. Here is Shneiderman's integrated portrait of DM:

- Continuous representation of the object of interest
- Physical actions or labelled button presses instead of complex syntax
- Rapid incremental reversible operations whose impact on the object of interest is immediately visible
- Layered or spiral approach to learning that permits usage with minimal knowledge

These features are not supposed to be a definition. They are a description in the context of 1982. Shneiderman was, of course, inspired by the shift from command line- to visual- and, finally, WIMP interfaces. However, we shall not overlook the generality of his thoughts. In particular, we should see the difference to WIMP, that we mentioned in the beginning. Neither does DM need to implement WIMP, nor does a WIMP interface necessarily comply to DM. The following modern reformulation by Apple Computer still captures the same notion [1]:

> "Direct manipulation allows people to feel that they are directly controlling the objects represented by the computer. According to the principle of direct manipulation, an object on the screen remains visible while a user performs physical actions on the object. When the user performs operations on the object, the impact of those operations on the object is immediately visible."

Another early example Shneiderman gave are display editors. Like Visicalc, their interfaces are character based. Still, they have a visual quality to them. That's why they are often called *visual editors*. Back then, these full screen editors were a spectacular improvement over line editors. Today, the species of character based text editors is almost extinct since modern graphical tools are way more convenient. The most prominent survivor is Vi on Unix systems.

However, for the most definite illustration of DM principles, Shneiderman points away from business software:

> "Perhaps the most exciting, well-engineered – certainly, the most successful – application of direct manipulation is in the world of video games."

Among his examples are some of the pioneering classics like Pong, Breakout, Donkey Kong and Space Invaders.

From all this observation, Shneiderman compiled the practical benefits DM has to offer:

- Learnability: Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user.
- Expert performance: Experts can work extremely rapidly to carry out a wide range of tasks, even defining new functions and features.
- Retention: Knowledgeable intermittent users can retain operational concepts.
- Less error messages: Errors are rarely needed to be handled with messages.
- Fast evaluation: Users can immediately see if their actions are furthering their goals, and if not, they can simply change the direction of their activity.
- Less Anxiety: Users experience less anxiety because the system is comprehensible and because actions are so easily reversible.
- Confidence: Users gain confidence and mastery because they initiate an action, feel in control and can predict system responses.

Once DM has been articulated and exemplified, its spirit and genius become intuitively comprehensible. The benefits listed above are beyond mere observation. They're more of a hypothesis. Still, they don't seem to call for a proof. We recognise them by our everyday computer interaction. Consequently, the challenge is to explain *why* DM seems so natural and desirable. We need an intellectual understanding of what DM is, how it works, when it works best and when it doesn't work at all. In the end, we want to be able to systematically think about user interfaces and fully exploit the benefits of DM. These questions motivated an academic debate that started at fundamental theory and later moved to the application in specific domains. In this seminar paper, we care about the first phase. Absorbing general aspects shall enable us to transfer and apply them to any of our own domains.

2. EXPLAINING EXPLAINATION

The Syntactic/Semantic Model

When Shneiderman initiated the debate about DM, he already contributed a model to explain it:

> "The success of direct manipulation is understandable in the context of the syntactic/semantic model."

He had developed this model of user behaviour earlier in the context of programming language experimentation. The model distinguishes *syntactic* and *semantic knowledge* in long term memory. What it thereby really utilises is a continuum that stretches from precise bits of action, like clicking a mouse button, to abstract meanings like the user's purpose in life. Shneiderman didn't invent this continuum, it plays a significant role in semiotics, computer science and linguistics. Nonetheless, by referring to the layers of syntax and semantics, he marks a point on it. In his model, *syntactic knowledge* comprises all knowledge about the interface that carries no meaning. In this sense, syntactic knowledge not only includes the grammar of the interaction language but also its vocabulary, morphology, phonology and phonetics. *Semantic knowledge*, on the other hand, is all knowledge which is independent of the interface, it may stretch to pragmatics and beyond.

Our continuum of abstraction is also a meter of relevance. More important than *how* to hit a target (syntax) is *what* the current target is (semantics), and even more important is *why* it was chosen (pragmatics). Now, we easily see the implication for interaction design: The machine is irrelevant. It's just a tool. What the user really thinks about is his goal- not his tool. Every moment he has to think about a tool is an interruption, and interruptions prevent flow. DM aims to support the user by lifting the interface to a more abstract and meaningful level. As the user has to deal less with the tool and more with his goal, the tool disappears. He is not sitting in front of a machine anymore but in the middle of his own domain (Figure 3). Under these conditions, the benefits Shneiderman claimed for DM, indeed, become self-evident.

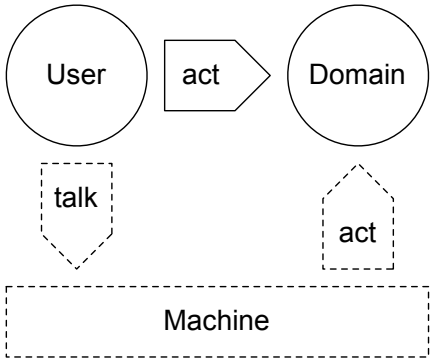

Figure 3. Direct- and indirect (dashed lines) interaction with the domain

Terminology

The message of DM certainly has a general core that still rings true today. We shall adapt the interface to the way the user thinks about achieving his goal. Yet, it also promotes a specific assumption that roots in the time when DM was conceived. In the seventies and eighties, applications were reenacting what, previously, had been done without computers like writing documents, editing tables and playing. It was assumed that the user thinks about achieving his goal in terms of real world objects. Consequently, the interface has to make these objects available. There would, at least, be some closely corresponding real world environment as an analogy. Because of this intended ostensible identity, the literature about DM neglects a crucial parting line that runs through terms like *metaphor*, *model* or even *domain*. So, let us translate these terms to our context:

1. A *domain* is an area of application. It refers to the user's high level goals, which are essentially independent of any interface and not necessarily object oriented.

2. A *model* is a pragmatic theory. It provides a simplified representation of a domain and enables us to think about the domain rationally. The model incorporates objects and rules. Two different models are relevant to us. One is the *mental model* that the user has in mind. The other one is the *interface model*.

3. A *metaphor* is an analogy to a model and may help representing it. Thus, the user may already be thinking about his domain in terms of a metaphor, even before being exposed to any interface.

To be clear: Mythology (metaphor) illustrates the condition (model) of human nature (domain).

Practical Concerns

The implicit assumption of classic DM is that the domain is already made of real objects so that it nicely maps to a model and hardly needs any metaphors to be represented. The table in VisiCalc is not a metaphor because it's not supposed to refer to anything other than a table. To the degree to that this assumption fails, interface- and mental model fall apart, and even more so do their metaphoric representations. The interface designer, as far as his creative leeway allows, tries to keep everything close together: domain, mental model, interface model, mental metaphor and interface metaphor. Shneiderman gave some advice on what is needed to design a DM interface:

- "appropriate representation or model of reality"
- "simple metaphors, analogies, or models with a minimal set of concepts"
- "representation must be meaningful and accurate to users"
- "careful design and experimental testing will be necessary to sort out the successful comprehensible approaches from the idiosyncratic ones."

However, the syntactic/semantic model already foretells some limitations the designer has to face. Here is our integrated and extended formulation of them:

1. A model and its metaphor often comprise no macro operations, so there is a trade off concerning the abstraction

level at which the domain is represented. If it's too low, micro management tasks are imposed on the user. If it's too high, certain operations can not be performed at all. Obviously, interface models tend to opt for functionality, which is why the first recipient of the Turing Award warned us of the Turing tarpit [2].

2. The representation of the domain can be misleading, since it is always just a representation- never the domain itself.
3. The interface doesn't help the user with problems that reside in the domain or his understanding of the domain.
4. As the interface adapts to the domain, it becomes domain dependent.
5. DM is not virtual reality. Users always must learn how the domain translates to the interface.
6. Domain conform visual navigation is often limited by screen space.
7. DM tends to demand more system resources.
8. Making actions traceable or reversible would often break the rules of model or metaphor.
9. The interface often has to be visual, which dismisses visually impaired users.

The basal limitation is, of course, the extent to which a domain can be represented faithfully. Yet, we often don't *want* to do that. A domain action may be cumbersome to represent, but it may also be a cumbersome action in itself. With DM, we model and represent the domain pretty straightforward, so we have to keep in mind that it isn't automatically handy. Its physics and organisation impose constraints. Often, the very purpose of the application is to perform actions that are, otherwise, impossible in the domain. Enslaving the interface to its domain or metaphor by faithfully representing *indirect* operations would certainly counter the intent of *direct* manipulation. It is a basic modelling task to dereference unnecessary indirections. As a simple illustration, take the cockpit view of a racing game. In the real world, we turn the steering wheel. In a game, we don't do that because what we actually want is to turn the car. Thus, designing a DM interface begins not with representing a model but with actively modelling the domain. Although, the user's mental model may not be particularly cumbersome, it may diverge from the interface model built by the designer. Now, who would have thought that there is actually some conflict between DM and user expectations? We'll return to this problem in the next chapter.

3. DIRECTNESS

In 1985, three academics were particularly dissatisfied with the explanations of DM that had been given, so they came together and published the next milestone paper on the topic [10]. Those three were Edwin L. Hutchins, James D. Hollan and, perhaps the most influential figure from the field of HCI, Donal A. Norman. They aimed for a cognitive account of DM to explain its advantages as well as its disadvantages. One of their basic assumptions was that the sensation of directness is the product of a number of factors, and that it must involve a cost and a trade off. They don't necessarily contradict Shneiderman but strive for more precision.

Cognitive Distance

We interpreted the syntactic/semantic model as a continuum from physical details in the machine to abstract meanings in the user's mind. On this continuum, DM shifts the machine closer to the user so that he has to deal less with low level issues. Hutchins et al. investigated the remaining distance. It is always up to the user to bridge the gap. He has to translate his goal to the interface language and backwards. In a way, he becomes the interpreter of his own mind. Because this cognitive effort is directed to the tool instead of the goal, it is experienced as a burden. Let us make this clear: We assume the user to be curious, creative and capable, but we acknowledge that thinking about meaningless matters annoys him. That is how Hutchins et al. understand directness as a quality that relieves the user of thinking. Or the other way around: distance = cognitive load. Thus, directness is not a property of the interface alone. It is the closeness between interface and user goals (Figure 4).

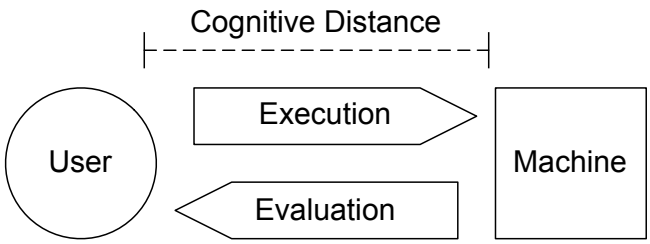

Figure 4. Indirectness as the cognitive effort of execution and evaluation

Further dissecting this cognitive distance, Norman introduced the *gulf of execution* and the *gulf of evaluation*. Few months later, in 1986, he published a book that eventually established both terms in the world of HCI [12]. The two gulfs pay tribute to the fact that input and output language of the interface may greatly diverge. Translating goals to machine input poses a different challenge than to translate machine output to a state of achievement.

Cutting the Distance

The second distinction about distance roughly corresponds to the syntactic/semantic model. As the list of limitations in Chapter (2) suggested, the user always has to translate his notion of the domain to the interface language, no matter how well the interface is designed. Limitation (1) particularly indicates that the interface model may not always be abstract enough for the user's high level goals. This part of the adaption to the interface relies on semantic knowledge so it accounts for *semantic distance*. At this point, the user still has to execute or *articulate* his specific intention through a physical input action. This adaption accounts for *articulatory distance*, which can be thought of as the *syntactic* side of interaction. Semantic- and articulatory distance add up to the total distance between user and machine (Figure 5).

Let's walk through a simple example. Suppose our goal is to get rid of a document. We bridge the semantic distance

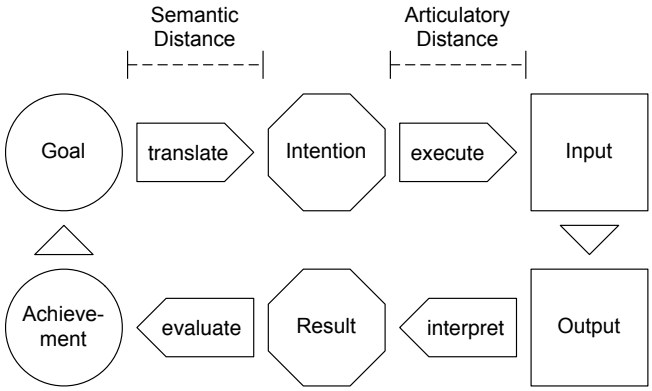

Figure 5. Semantic- and articulatory distance at both gulfs of interaction

by realising that the office that our system somehow represents incorporates the notion of a trash can. We translate our goal to the intention of throwing the document into the trash. Now, we bridge the articulatory distance by using the mouse to drag the document icon onto the trash can icon. We experience great articulatory directness. Firstly, because the physical movement of our hand correlates perfectly with the intended movement of the document on screen and, secondly, because the trash bin is now filled. However, the experienced semantic directness depends on our mental model of the office. If our first instinct was to rip the document apart and forget it, then remembering the bin and evaluating whether the document has really gone was a burden.

Interestingly, the example continues the point that we made at the end of the last chapter. Even if we had immediately anticipated the trash and, thus, experienced more semantic directness, the action would have been cumbersome. We would have shared our model with the model of the interface, but because this model is so obedient to the office domain, it isn't as direct as it could be. The spirit of DM demands of the designer to teach us what we want- not to ask for it.

Figure (6) summarises the relation between some of the layers that we discussed until now.

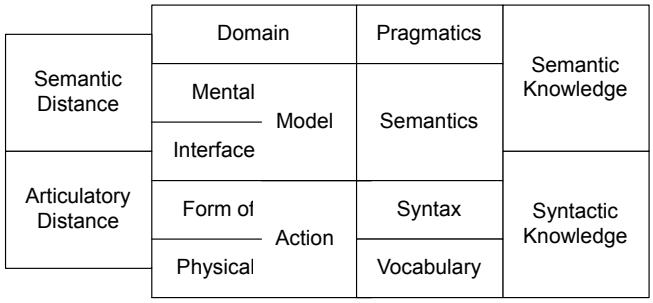

Figure 6. From cognitive distance back to the syntactic/semantic model

Articulatory Distance

Hutchins et al. were well aware that input- and output devices determine the limit of articulatory directness:

> "The restriction to simple keyboard input limits the form and structure of the input languages and the restriction to simple, alphanumeric terminals with small, low-resolution screens, limits the form and structure of the output languages."

They continued:

> "Increasing the articulatory directness of actions and displays requires a much richer set of input/output devices than most systems currently have."

Since these statements were published in '85, countless technological innovations have facilitated articulatory directness. The greatest boost certainly came from touch screens. Semantic directness, on the other hand, remains a design issue that is getting more and more complex, as the articulatory interaction languages get richer.

Semantic Distance

We identify two fundamental approaches of handling semantic distance: *conformity* and *convergence*. Let us explicate them in a little more essayistic style and even risk taking sides.

Conformity

The first challenge for the interface designer is to model the domain for semantic directness and balance the trade off mentioned in Limitation (1), while also conforming to the user's mental model. Since this is nearly impossible to achieve, he focuses on the second challenge, which is to find a representation that is at least compatible to any metaphor the user might have applied to his mental model.

The challenge for the user is literally to get used to how the interface represents the domain. To a skilled user, the interface may feel more semantically direct because he automated mediating steps thereby regaining some flow and autonomy. Though they admit that automation doesn't actually reduce semantic distance, Hutchins et al. state:

> "The frequent use of even a poorly designed interface can sometimes result in a feeling of directness like that produced by a semantically direct interface."

We want to argue that effort reduction through automation always feels problematic, at least on a subconscious level because the actions taken out are less meaningful. A psychopath, at some point in his childhood, has separated his actions from his motivations in order to conform to some authority [9]. Just like that, the long time user of a poorly designed interface, at some point during his adaption, has separated his intentions from his goals in order to conform to the system. One practical problem is that he gets dependent on the interface and on his own set of tasks. Whenever one of both changes, he has to dig down again to the low level elements of which his automated behaviour was once constituted. Also, a cumbersome interface is seldom the best one available. For every command line tool, someone has built a high level visual interface. That's what freedom and competition do. Nowadays, users intuitively understand how fast that dynamic works and only bother to get into a product if

it's as concise, elegant and close to their domain as can be. In contrast, learning a poorly designed interface by root memorisation and thinking of that as expert knowledge is what psychopaths do.

Convergence

Conventional design gets us only so far, and practice is not a real option. However, there is a third way, and it poses a task to the designer as well as the user.

An interface can not match the user's mental model all the time. One reason are the limitations discussed in Chapter (2). Another one is the mental model itself. So, what is a mental model anyway? It's what the user has learned. It certainly isn't an accurate account of the domain. Instead, it's simplified, distorted, fuzzy and amorphous. Moreover, it depends on the user's life experience, education and cultural background. So, instead of guessing his thoughts, what if the interface could shape them? What if the user would be naturally taught to adapt his conceptualisation of the problem so that he comes to think of it in the same terms as the interface? In any case, the application is supposed to provide a powerful way to think about the domain. When it also teaches that, the user's mental model is guaranteed to be a consistent match and predictor of the interface model.

Now, is it still appropriate to try to anticipate the user's notion of the *real world*? Are real world metaphors generally accessible? There is another aspect to that. The world and the machine are blending, and if it isn't simply becoming invisible, the machine becomes the world. As we're assimilating the nature and ubiquity of applications, the need for analogy vanishes, and metaphors get displaced by mascots. A *cloud* doesn't make us think of rain anymore. At the same time, it doesn't explain anything. It doesn't have to. The idea of networked servers and clients is all too familiar.

Also, computers evolve at such a speed that they already devour external metaphors. Think of those taken from storage mediums: film, tape, disc or book. Young people have more experience with the digital applications than the physical objects. The interface explains to them its metaphor, instead of the metaphor explaining the interface. Furthermore, we're so overwhelmed with information and concepts that digital data loses its role as a tool and becomes the domain itself.

There is only one direction. The *real world*, which means *the world that doesn't have an interface*, won't keep up with technological and conceptual transformation. It won't be the place of many domains, and it won't provide meaningful metaphors. Since interface designers create much of our experience, it is up to them to provide genuine models and representations, even if those inventions start out as references to themselves. Hutchins et al. get to the same conclusion:

> "But if we restrict ourselves to only building an interface that allows us to do things we can already do and to think in ways we already think, we will miss the most exciting potential of new technology: to provide new ways to think of and to interact with a domain."

Finally, we might solve the intricate trade off that we frequently referred to. As pointed out by Limitation (1), the designer is forced to either sacrifice semantic directness for functionality or to mess up model and metaphor. However, if he is the one who creates both, he may not just integrate macro operations in a low level model but design the model from the outset so that it scales well with the user's goals.

Semantic distance plays a significant role in applications for software development. That is why the syntactic/semantic model stems from the context of programming languages. The idea of scalable models really became manifest in functional programming languages like Lisp. Paul Graham wrote [8]:

> "In Lisp, you don't just write your program down toward the language, you also build the language up toward your program. [...] once you abstract out the parts which are merely bookkeeping, what's left is much shorter; the higher you build up the language, the less distance you will have to travel from the top down to it."

A Lisp programmer has neither a micro management problem nor does he miss functionality. That principle is exactly what we need to reduce semantic distance. The model should incorporate an innate scalability so that, from within the model world, the user can naturally choose the precise abstraction level that he deems interesting.

The problem with the third way is that it's hard. In order to create a powerful scalable tool, the designer has to be more involved in thinking about the domain model. In order to apply that tool, the user has to learn a new way of thinking about his goal. In order to converge, both must be willing to think differently. Another recipient of the Turing Award, E. W. Dijkstra, said about the untapped potential of simplification in software design [4]:

> "Firstly, simplicity and elegance are unpopular because they require hard work and discipline to achieve and education to be appreciated."

Very much in the sense of that statement, designers and users opting for the way of *convergence* face initial difficulties that *conformity* simply doesn't bring up. The benefits, though, might be huge.

4. MANIPULATION

So far, we've only elaborated on directness, but why didn't Shneiderman just speak of *direct* interfaces? Does a small distance not imply a *direct manipulation* interface? Apparently, it doesn't.

Engagement

Directness, as previously described, only implies minimal cognitive load. Selecting a menu option, completing a web form or googling can all be very direct interactions, but they normally don't count as DM. We'd have to try hard to see them as manipulations of interesting objects. We may also notice how they lack a certain quality of *engagement* that would spread from playing around with objects.

With this kind of reasoning, Hutchins et al. establish engagement as a distinct quality of interaction that is independent of directness. They attempt to span a two-dimensional *space of interfaces*. Yet, we may notice that its dimensions are somehow correlated. The lack of engagement may simply be interpreted as semantic distance. If the user needs to manipulate interesting objects to experience the interaction as less of a burden, then he longs for less cognitive load. Thus, the distinction between small distance and engagement seems quite fuzzy. Hutchins et al. even express their explanation of engagement in terms of directness:

> "Direct engagement occurs when a user experiences direct interaction with the objects in a domain. Here there is a feeling of involvement directly with a world of objects rather than of communication with an intermediary. The interactions are much like interacting with objects in the physical world. Actions apply to the objects, observations are made directly upon those objects, and the interface and the computer become invisible."

How is this different from what we already learned about distance? The authors go on to list four minimum requirements of *direct engagement*, which mostly describe directness. The first requirement even explicitly states:

> "Execution and evaluation should exhibit both semantic and articulatory directness."

So, if directness is a *precondition* of engagement, how can both be independent enough to span a space? We'll try to sort this out in the following.

Conversation

What Hutchins et al. were getting at is to contrast manipulation with conversation. We already mentioned the example of deleting a document. The conversational approach is to mark the document and select the menu option that says *Delete*. This isn't even purely conversational since we still identify the document by simply pointing to it. However, we don't throw it away by ourselves, instead we utter a command to the system, which, in turn, tells us how the operation went. The interface represents a partner with whom we have a conversation about an assumed model world.

Hutchins et al. came to understand engagement and manipulation as the absence of conversation. By factoring out engagement, they restrict the meaning of *semantic directness* to whether the interface model is abstract enough for the mental model. At this point we also have to recap that the problem of cognitive directness is mainly a problem of semantic directness. Having said all that, completing a web form, selecting an option or following the instructions of a software wizard can, indeed, feel very direct, and the *space of interfaces* (Figure 7) makes a little more sense to us.

The opposite of high level conversation with a wizard is low level manipulation of a model world. Both variants do not count as DM but may be appropriate for certain applications.

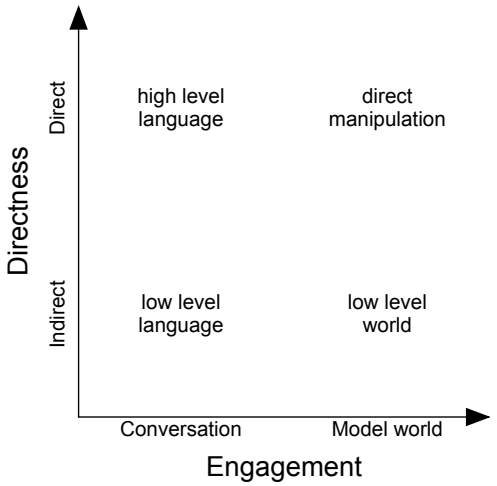

Figure 7. Interface space according to Hutchins et al. (We inverted the vertical axis.)

5. MIXED MODE

Hutchins et al. enabled fundamental insights into the anatomy of DM. In terms of our understanding, it is a precise idea, but in terms of scientific evaluation, it is rather vague and hard to operationalise. The potential impact of each benefit claimed by Shneiderman depends on the domain, the user and on how DM principles are implemented. Different studies rejected different benefits so research questions had to become more specific: For a given domain, which aspects of DM result in which benefits? These questions led to a greater appreciation of the fact that practical interfaces often complement manipulation with conversation. Desktop operating systems mix a model world of folders and documents with the conversational approach of menus, dialogs and wizards. Probably, they do it for good reason.

Manipulation vs. Conversation

We came to think of conversation as a cumbersome mode because it is conceptually indirect (Figure 3). However, the paradox is that conversation sometimes *feels* more direct than manipulation, which led to a debate about how contrary both interaction modes really are.

Before entering this debate, we need to point out that the literature still tends to take *direct manipulation* for *manipulation* although the interface spaces make a clear distinction there. We don't dive into the details of all the criticism that has been lavished on DM because much of that only applies to the *manipulation* aspect and strives to make a case for conversation. Instead, we proceed with our attempt to integrate different theoretic perspectives and, thereby, free the DM philosophy from misconceptions.

From early on, Shneiderman was aware of the superiority conversation can have over manipulation. In 1982 he remarked:

> "Knowledgeable and frequent users prefer command languages because in many cases they permit faster task completion."

Still, Shneiderman participated in the ongoing debate arguing on the side of manipulation [16].

At the Personal Systems Laboratory of Hewlett Packard, David M. Frohlich conducted intense research on the matter [5, 6]. In 1993, he noted that conversation "leads to a language-based form of interaction which exploits the instrumental and regulatory functions of language ..." Considering directness, he said:

> "In fact, the entire debate about the relative advantages and disadvantages of language versus action based interfaces turns on an attempt to explicate the conditions under which each is most direct."

In an attempt to integrate both interaction modes, he proposed a social definition of directness. According to Frohlich, conversation induces three kinds of indirectness. We identify them as *semantic distance*, *convention* and *implication*.

1. *Semantic distance*
   "... cognitive effort involved in describing certain concepts in language." This indirectness should correspond to the semantic distance of the conversational interface. It is especially high when the intentions involve spatial information, like when explaining directions over the phone.

2. *Convention*
   "... conventions of politeness [...] These kinds of inhibitions and constraints appear to be relaxed with increases in rapport between parties ..." If this is, at all, relevant to interaction design, then only at the gulf of evaluation.

3. *Implication*
   Participants of a conversation "manage to mean much more than they actually say [...] they rarely say directly what they mean."

On the basis of experiments conducted by Garfinkel in 1967 [7], Frohlich argued that *implication* had previously been underestimated in the context of interfaces:

> "All this is a puzzle for the original notion of direct interaction in HCI, since we have now described a pervasive form of *indirectness* in conversation which seems to underlie highly efficient interaction. Furthermore, its very efficiency appears to rely on *more* rather than less cognitive work being performed by the interactants."

In order to better relate conversation to DM, we have to understand *directness* more accurately in terms of *efficiency*. We don't need to discard the idea that indirectness equals cognitive load, like Frohlich does. We just have to grasp the generality of *cognitive load*. It is not about an absolute amount of effort but about what we're getting for it. The additional effort of dealing with a conversation partner pays off when this partner executes our high level commands. Instead of searching for the trash bin, we right-click the document and tell the system "Move to Trash". Instead of repeating a manual action, we write a script of instructions for the system. Through *implication*, conversation can be more efficient and, thus, more direct. To make this point, Frohlich introduced additional terms like *social directness*, *graceful-* and *clumsy interaction*. Figure (8) illustrates how he accommodated the space of interfaces.

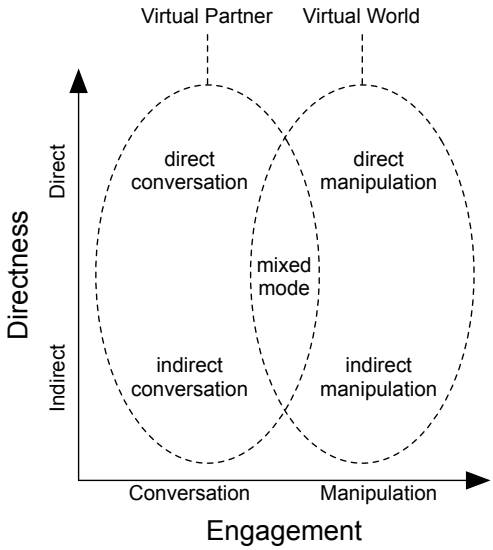

Figure 8. Interface space according to Frohlich (We adjusted the labels of the vertical axis.)

So, what is the big implication for us? First of all, Frohlich confirmed that DM is just one area in a space of interface types. Second, manipulation isn't always the most appropriate mode because conversation can be more efficient. Third and most importantly, an interface may have to *combine* manipulation and conversation to hit the sweet spot of usability. Mixed mode interfaces are especially needed for complex applications that incorporate visual and non-visual tasks. Manipulation alone would be more about interactive *information visualisation* [6]. To advocates of pure manipulation this means that it may have to be applied selectively to a more specific aspect within an application. As early as in 1993, Frohlich concluded quite philosophically:

> "Fortunately there is a symmetry to these conditions which means that language is often well designed to overcome the problems with manipulation as a method of interaction, and vice versa. This complementarity of physical and social activity should not be seen as an accidental feature of system design but rather as a deep property of human collaboration and perhaps even of life itself."

Agents

Of course, the consensus that most interfaces are – and probably should be – *mixed mode* interfaces didn't end the discussion. After all, they present the designer with a dualism that he still has to balance. So, how shall manipulation and conversation blend? How shall the system be represented as a partner of conversation? The obvious answer would be *personification*. In its most consequent implementation, the interface represents the system as an agent, which is a virtual character with whom the user may communicate.

The alluring quality of agents is that they may overcome the

barrier between conversation and manipulation. An agent can be presented as an object that resides and acts within the model world, just like an adversary belongs to our mental model of playing table tennis. In this sense, conversation would just be a form of DM. We would manipulate the agent object by feeding it language expressions.

On the other hand, agents have proofed to be somehow difficult. As we might suspect by now, interfering with the model and its metaphor provokes problems, but there is also a more fundamental reason. Shneiderman already hinted that the representation of real objects may be misleading. His warning applies even more to objects that are supposed to behave socially. A computer can easily visualise a character, but it cannot adequately reproduce social interaction. In 1970, Masahiro Mori had coined the term of the *uncanny valley* for this mismatch of appearance and behaviour [11]. Frohlich commented in 1993 [5]:

> "The growing debate about interface agents seems to turn on the extent to which this virtual partner should be characterised explicitly at the interface."

Ever since the Xerox Star machine started a revolution, operating- and office systems have determined and reflected what kind of interface users expect. Today, it seems that the opposite approach to agents has come out on top. Microsoft tried to establish agents when it introduced the Windows search dog and the Office assistant, but both were widely perceived as annoying and have been removed from their products. The conversational side of interfaces sticks to menus, dialogs, scripts and the like. A more deliberate step toward natural conversation has recently been taken by Apple Computer. Although Siri provides a sophisticated personal assistant to whom the user can actually speak, it is modestly represented as what it ultimately is: a microphone. In the end, we primarily communicate with a system and only secondarily manipulate a model world. The physical system gives rise to a virtual model- not the other way around.

6. DELVING INTO SPACE

What we find most striking about Frohlich's insights is that, in our view, they pave the way for a space of interfaces that makes more sense than the one suggested by Frohlich himself. Let us explain this in detail.

Logical Black Holes

Much debate and confusion stems from the ambiguity of the term *directness*. We identify two fundamentally different meanings.

1. Figure (3) illustrates *conceptual directness*. The interaction is conceptually direct if the user is not delegating actions to the machine but is himself engaged in acting on the domain. The resulting quality is *engagement*. To be engaged also means to see and understand what is being done. It is intrinsically linked to activities that are difficult to assess in terms of efficiency like learning, exploring and playing. Manipulation is the interaction mode that seamlessly translates to conceptual directness. It appeals to the senses and can literally bring domain objects to the user's mind.
2. The second kind is *practical directness*. The interaction is practically direct if the user can express his specific goals to the machine without cognitive effort. Frohlich recognized that conversation feels direct because it's efficient, but he didn't adapt the definition of cognitive directness, which led him to split it in two types. Practical- is basically the same as cognitive directness, but we emphasise the point that, from the very beginning, it was meant to be *all* about *efficiency*. The user wants to transmit a lot of information to the system at the cost of little effort. The natural way to do this is the natural way in which he exchanges information: through conversation. Because the building blocks of conversation (not necessarily language) are informative (implication), abstract and flexible, they enable the user to express his (high level) thoughts efficiently.

In summary, some important insights follow from all the debate about directness, manipulation and conversation.

1. *Manipulation* means the natural way towards engagement.
2. *Cognitive directness* means efficiency.
3. *Conversation* means the natural way towards efficiency.
4. Great user experience demands engagement *and* efficiency.

Acknowledging these insights enables us to expose *why* the proposed interface spaces (Figures 7 and 8) seem somehow artificial or wrong. Note that *directness* in these spaces and most other contexts means cognitive- (practical-) directness.

1. When directness and engagement are portrayed as independent dimensions, conversation can't be the absence or converse of manipulation, instead it must correspond to directness.
2. A little engagement with a little directness should, in principle, result in the same type of interface as high engagement with high directness. The type at the bottom left just has less of both dimensions and is worse than the one at the top right.
3. Since engagement and directness are both desirable, the optimum interface type should be at the top right. Instead, Frohlich puts mixed mode interfaces in the middle.
4. Having mixed mode interfaces below directness and to the left of manipulation suggests that there can be *too much directness* or *too much engagement*.

The Depth of Simplicity

Now, we can resolve the repugnancy of the interface spaces we discussed and get to a better understanding of how mixed mode relates to DM. Figure (9) shows the interface space that we envision.

So, where is *direct manipulation* in our space? Shneiderman's attempt to label the ideal interface type just wasn't as subtle as the understanding at which we arrive today, or maybe, he wasn't as distracted by dubious subtleties. While *directness*

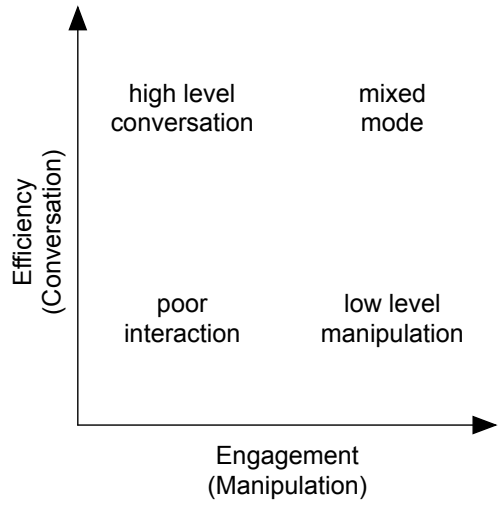

Figure 9. Our space relies on an unmitigated notion of directness.

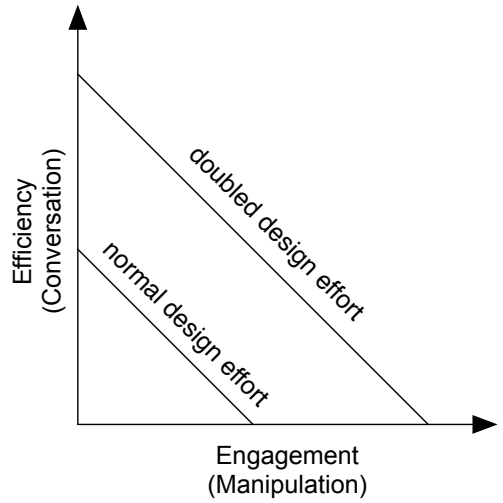

Figure 10. The weighting of design goals is independent of total effort.

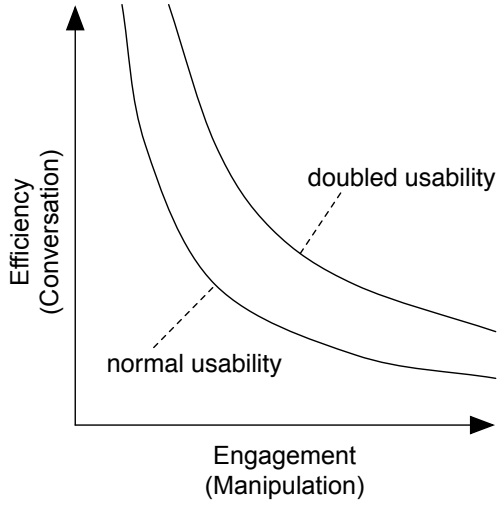

Figure 11. Usability as a product of efficiency and engagement.

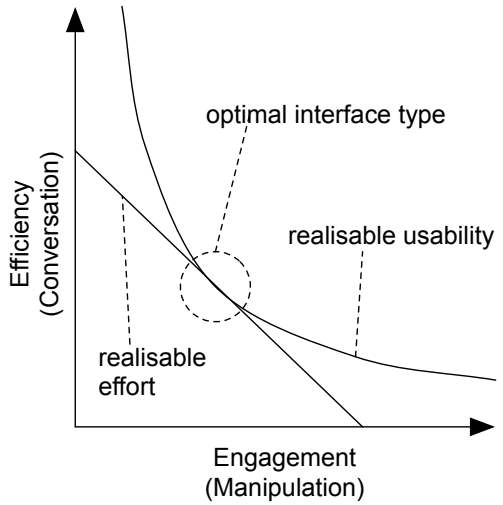

Figure 12. Maximising usability for the realisable amount of total effort

can mean the general design goal, *manipulation* is a particular interaction mode that promotes conceptual directness (engagement). Through declaring *directness* as practical directness (efficiency), *mixed mode-* and *direct manipulation* interfaces are revealed to be the same thing. We conjecture that, from the very beginning, DM was meant to refer to mixed mode interfaces as we understand them. In that sense, the holy grail remains where efficiency meets engagement.

Apparently, there is a trade off between efficiency and engagement. Both have to share the focus of the design process and are somehow contrary as pointed out in Limitation (1). Figure (10) illustrates what types of interfaces can be achieved at two different levels of quantitative effort.

However, the co-dependence between efficiency and engagement runs deeper. For a certain standard of usability, the designer can not easily substitute one design goal for the other. We may explain this in terms of micro economics [3]. Efficiency and engagement don't just add up to usability, instead they are *factors*, and usability is their *product*. All factor combinations that produce the same amount of the product are lying on a convex isoquant. The highest amount is achieved where both factors are high. Look at Figure (11) to see how interface types relate to usability standards.

The challenge for the designer is to really hit the sweet spot between conversation and manipulation instead of just investing more time and effort. It doesn't matter to this principle what other design goals we might cram into the term *usability*. Figure (12) finally shows how a balanced mix of interaction modes maximises the quality of user experience.

Integrating Outliers

There is one last thing. Our space inherits two freaky interface types. In the spirit of Hutchins et al., we labelled them *high level conversation* and *low level manipulation*. With Figure (12) in mind, we might wonder why anyone would need an interface type far to the left top or right bottom on the

effort line or in the whole space, respectively. We start explaining this by reconsidering why conversation is associated with high level- and manipulation with low level interfaces.

The abstraction level of the interface is always relative to the user's goal, and that goal is always more abstract than the interface model. No application has that *make me happy button* that does it all. There is always a little cognitive distance to walk down to lower level entities. Why down? Because if the user would have to walk up, the application would be of no use to him. If he has to talk in pre-built sentences and is not allowed to arrange the words by himself, he can't express his goal. So applications approach the user from the complexity below him.

Denying access to low level objects decreases engagement, while denying high level commands decreases efficiency. Providing all those levels is hard work on side of the application designer, so he estimates on which the user wants to operate. The farther away an abstraction level is from the user, the smaller is its contribution to usability, and the more expensive it becomes for the designer to increase usability in that direction. The way Figures (10, 12) display our space implies that efficiency and engagement come at equal cost because each effort line (isocost) encloses an equal sided triangle with both axes. Note that the designer chooses the location of the isocost (his absolute effort) but not its slope. The slope reflects the relative costs of engagement and efficiency.

When the user only needs a few high level operations, it's cheapest to design for efficiency and the isocost is more vertical. When the user enters the stage at an already complex level, engagement is cheaper and the isocost more horizontal (Figure 13). Mixed mode interfaces like in Figure (12) are made when the user likes to roam at many levels.

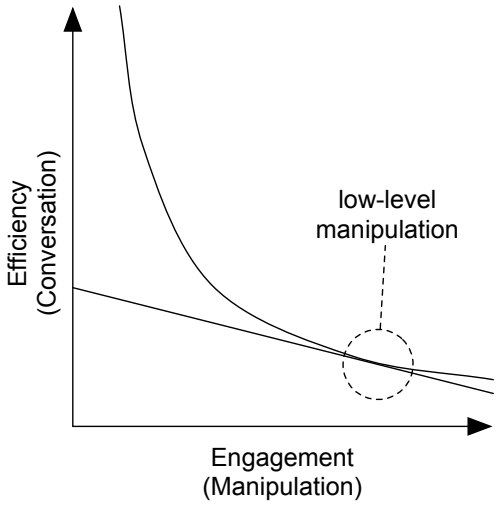

Figure 13. One mode may still be more effective (cheaper) than the other.

The designer can not build high level functions without lower level building blocks, but he can provide low level objects without topping them off. Opening the basement of a sky scraper of which the user only needs the top level poses a smaller risk than building a sky scraper of which the user only visits the basement. Just providing the low level is also *easier* than anticipating the user's goals and building a hierarchy up towards them. That is the reason why agents are so problematic, while low level manipulation can be found everywhere. Engagement tends to be cheaper, not only for complex applications.

In the most extreme case of low level manipulation, object oriented design is practically omitted, and the interface model has to reflect the domain in all its glory and complexity without adding expensive higher levels. The most radical advocate of this finding is Richard Pawson, who established the architectural design pattern *naked objects* [13]. Actually, naked objects not only determine the architecture of an application but also the interface, which is generated generically from a strictly domain driven model. Naked objects have been successfully applied to dramatically complex domains. We observe, once again, how meaningful interface design starts with modelling. Or in the words of Steve Jobs [17]:

> "It's not just what it looks like and feels like.
> Design is how it works."

7. CONCLUSION

By pointing out the principles of DM, Shneiderman also pointed academic awareness towards human factors of software interfaces [14, 15, 16]. Major contributions to conceptualise human-computer interaction came from Hutchins et al. [10], Norman [12] and Frohlich [5, 6].

We learned to distinguish semantic- from syntactic knowledge, the gulf of execution from the gulf of evaluation, articulatory- from semantic distance and conformity from convergence. We came to conclude that the interface designer has to be bold and start the design process with modelling the domain in a way that makes the model intrinsically adaptable to the user's way of thinking. He also should be ambitious enough to teach the user instead of guessing the user's knowledge.

We further deducted a fundamental trade off that seems to underly all interfaces and is deeply connected to the core dualism and conflict of human nature:

interaction quality	efficiency	engagement
interaction mode	conversation	manipulation
kind of directness	practical	conceptual
user experience	getting things done	experience the process
focus of attention	abstract goal	physical object
cognition	thought	sensation
medium	(formal) language	visuals, sound
legacy	culture	nature
explaination	psychology	biology
typical tasks	collaborate, retrieve, calculate, compile, automate	navigate, explore, experiment, compose, play

Balancing this dualism is the basic task of designing an application. The best interfaces tend to mix interaction modes,

although it seems to us that the side of engagement is kind of a basis on which the other side relies. When efficient conversation is too hard to achieve, the interface can (and often does) fall back to engaging manipulation.

It remains unclear how this balance differs between the gulfs of interaction or between semantic- and articulatory distance. It may be the case that conversation is more dominant at the gulf of execution, while manipulation is more important to the evaluation side. Such questions are beyond the scope of this work, but the insights we gained already made us think differently about interaction design.

REFERENCES

1. Direct manipulation. `http://www.infovis-wiki.net/index.php?title=Direct_manipulation`, January 2013.
2. Turing tarpit. `http://en.wikipedia.org/wiki/Turing_tarpit`, January 2013.
3. Breyer, F. *Mikroökonomik*. Springer-Lehrbuch. Springer, 2005.
4. Dijkstra, E. W. The next fifty years. `http://www.cs.utexas.edu/users/EWD/ewd12xx/EWD1243a.PDF`, 1996.
5. Frohlich, D. M. The history and future of direct manipulation. *Behaviour & Information Technology 12*, 6 (1993), 315–329.
6. Frohlich, D. M. Direct manipulation and other lessons. In *Handbook of human–computer interaction (2nd ed)*, Elsevier (1997), 463–488.
7. Garfinkel, H. *Studies in ethnomethodology*. Prentice-Hall, 1967.
8. Graham, P. *On Lisp: advanced techniques for Common Lisp*. Prentice Hall, 1994.
9. Gruen, A. *The Insanity of Normality: Toward Understanding Human Destructiveness*. Human Development Books, 2007.
10. Hutchins, E. L., Hollan, J. D., and Norman, D. A. Direct manipulation interfaces. *Hum.-Comput. Interact. 1*, 4 (Dec. 1985), 311–338.
11. Mori, M. The uncanny valley. *Energy 7*, 4 (1970), 33 – 35.
12. Norman, D. A., and Draper, S. W. *User Centered System Design; New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1986.
13. Pawson, R., and Matthews, R. Naked objects: a technique for designing more expressive systems. *SIGPLAN Not. 36*, 12 (Dec. 2001), 61–67.
14. Shneiderman, B. The future of interactive systems and the emergence of direct manipulation. *Behaviour and Information Technology 1*, 3 (1982), 237–256.
15. Shneiderman, B. Direct manipulation: A step beyond programming languages. *Computer 16*, 8 (aug. 1983), 57 –69.
16. Shneiderman, B., and Maes, P. Direct manipulation vs. interface agents. *interactions 4*, 6 (Nov. 1997), 42–61.
17. Walker, R. The guts of a new machine. `http://www.nytimes.com/2003/11/30/magazine/30IPOD.html`, November 2003.