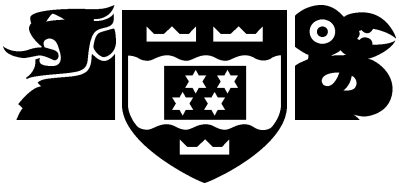
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@mcs.vuw.ac.nz

User-Interface Metaphors in Theory and Practice

Pippin Barr

Supervisor: Robert Biddle

December 8, 2003

A thesis

submitted to Victoria University of Wellington
in fulfilment of the requirements for the degree of
Master of Science in Computer Science.

Dedicated to the pigeons and sparrows of Wellington.

Abstract

User-interface metaphors are a widely used, but poorly understood, technique employed in almost all graphical user-interfaces. Although considerable research has gone into the *applications* of the technique, little work has been performed on the analysis of the concept itself. In this thesis, user-interface metaphor is defined and classified in considerable detail so as to make it more understandable to those who use it. The theoretical approach is supported by practical exploration of the concepts developed.

Contents

List of Figures

Chapter 1

Introduction

In the following thesis the concept of using metaphors in the user-interface to explain the functionality of computer systems is investigated. The approach taken is to apply analytical techniques from linguistic philosophy and semiotics to the metaphors found in the user-interface. After developing a detailed theory of user-interface metaphors, a move toward practical applications is also made.

1.1 Metaphors in the User-Interface

The concept of a user-interface metaphor is one which is widely known in the field of human-computer interaction and often utilised in the process of user-interface design. In the past, user-interface metaphors have been thought to be the ideal solution to helping first-time and novice users to use computer software immediately, with a minimal learning period. Because of this proposed benefit, user-interface metaphors were and still are used a great deal in most interfaces [7, 38, 39, 45, 48, 77].

More recently, doubts have been voiced about the utility of metaphor in the user-interface. Certain studies have suggested that metaphor is *not* as useful as it has been claimed, and, overall, the level of enthusiasm about the concept has dropped considerably [14, 40, 44, 54, 64, 81].

One of the most prominent problems with the use of metaphor in the interface is the general lack of understanding of the fundamental concept itself. That is, for the large part, designers do not necessarily have a deep grasp of what metaphor is, and how it behaves, especially when used in an interface. It is possible that this lack of understanding might be holding back truly useful interface designs.

A key issue in the understanding of user-interface metaphors is simply that the general concept of metaphor is not necessarily well understood by designers, or at least has no generally accepted definition or model attributed to it. Given this, it is a difficult task to understand user-interface metaphor with any certainty or rigour.

This leads designers to believe that the concept of a user-interface metaphor is extremely simple, and is limited to something along the lines of "*using a real world concept to help the user understand how to use computer software.*" It should be clear that, while this does characterise the very *basic* idea of metaphor in the user-interface, it does not offer any explanation as to how this process works, or expose the concept in any depth whatsoever. Using a definition such as just stated leaves user-interface designers in the position of having to rely solely on intuition in order to design metaphors for explaining software. Intuition is a useful aid, but it is likely not adequate as the *only* means for designing a user-interface metaphor.

Although it was stated above that metaphor is a concept that is difficult to understand in *general*, it is also the case that a considerable amount of analysis has already been attempted. Particularly, the philosophy of metaphor is a large and deep area, which has provided several detailed theories of the inner workings of metaphor as well as attempts to classify and identify the occurrence of metaphor in language. Semiotics is another field which has seen some analysis brought to bear on the concept of metaphor. Thus, while there is not necessarily a definitive answer to the questions of what metaphors is and how it works, there has been plentiful research into the matter.

The state of analysis of metaphor in philosophy, linguistics and semiotics is considerably further advanced than what is available for the understanding of user-interface metaphors. Recent times have thus seen efforts to apply this "outside" research to the problem of metaphor in the interface, and this seems a valuable objective. One aspect of non-user-interface research that would appear particularly valuable is the analysis of exactly what metaphor *is* and how it works, as well as the sorts of things it can be used to achieve. Defining and analysing metaphor more deeply is the subject of this thesis.

In this thesis the work of Lakoff and Johnson developed in *Metaphors We Live By* will form the focal point of understanding the ways in which metaphors works, as well as its typical uses [43]. The book presents a detailed account of the working of metaphor in every day human language and cognition, and also suggests ways in which metaphor can be generally categorised and identified. This kind of research will be shown to be highly applicable to the concept of user-interface metaphor, and, it will be claimed, provides significant insights into the concept. In addition to the work of Lakoff and Johnson, a semiotic model of user-interface metaphor will be explored largely using the work of Charles Sanders Peirce [67] and Umberto Eco [27]. This model is intended as a first attempt to characterise the underlying structure of a user-interface metaphor in order to provide a means for talking about the concept in detail. The model will also provide a means for analysing the metaphor design process itself, from conception to implementation and actual use in.

1.2 Structure

The structure of the thesis proceeds from a basic introduction to the concept of the user-interface metaphor to a means of classifying metaphors and analysing their inner structure. The thesis is completed by practical applications of the concepts developed.

Chapter 2 provides a very basic introduction to the concepts of metaphor and user-interface metaphor as well as an extensive literature review of the research already undertaken in this area. The chapter emphasises the need for a detailed assessment of the concept of metaphor itself, rather than proceeding directly to the analysis of its use in practise.

Following this, chapter 3 embarks on just such a task by applying the work of Lakoff and Johnson in *Metaphors We Live By* to the user-interface metaphor. This chapter provides descriptions of the Lakoff and Johnson material, followed by detailed explanation of how it can be applied to the user-interface in a principled manner. This work focuses primarily on the ability to use the classifications of metaphor provided by Lakoff and Johnson on user-interface metaphor in order to gain insight into the concept. The chapter thus demonstrates in some detail how research from other fields can be successfully transfered to the consideration of the user-interface metaphor.

Having shown how Lakoff and Johnson's work can be applied to the classification and understanding of user-interface metaphors, a complementary model of their inner structure is provided through semiotic analysis in chapter 4. This chapter introduces the concept of semiotics and semiotic models in some detail before developing models of metaphor and user-interface signs in turn. Next, the two models are combined to provide a detailed analysis of

the structure of user-interface metaphors as well as an account of their overall design process. The chapter aims especially to provide a consistent terminology for use when discussing user-interface metaphors, as well as a structured understanding of their inner-workings.

The work on semiotics concludes the detailed theoretical component of the thesis and so chapter 5 provides an overview of the most important points made in the form of a taxonomy. The taxonomy structure is based partly on the concept of interaction design patterns which are a promising application of the concept of design patterns in general [30, 83].

In chapter 6 the taxonomy is applied to a real user-interface in order to test the concepts contained within it. This involves using the classificatory system developed in chapter 3 as well as the semiotic model of chapter 4 to gain insight into the working of the Project Gallery interface in Microsoft Office.

Following this, a set of heuristics is developed based on the theoretical material in the thesis as well as the practical work in the case study. The heuristics are presented in some detail in chapter 7, including general discussion of their derivation and use, as well as examples.

Chapter 8 finishes the research in the chapter by describing a heuristic evaluation performed using the heuristics of chapter 7. This chapter shows that the heuristics can be properly applied to a real interface and can elicit useful insight. More importantly, considerable discussion of *improving* the heuristics and the evaluation process is provided along with thoughts as to future work to perform in the area of heuristic evaluation.

Having described the theoretical research as well as its practical applications, the thesis is concluded in chapter 9. After analysing the overall structure of the thesis, some indications of further work are suggested before the several concrete contributions of the work are explicitly noted.

Chapter 2

Metaphor and Human-Computer Interaction

2.1 Introduction

This chapter establishes the place of metaphor in the field of human-computer interaction. This means, first of all, understanding exactly what metaphor is and what it does, which is examined in section 2.2. With this in mind, in section 2.3, it is shown that metaphor is used pervasively to help us understand computing in general. In section 2.4 the focus is narrowed to the aspect of computers that this thesis concerns: the user-interface. In particular, a definition of the concept of a *user-interface metaphor* is provided. Section 2.5 introduces some of the academic work on user-interface metaphors. This section outlines some of the critical evaluation of the idea, as well as indicates some of the more practical work toward helping designers *design* user-interface metaphors. Finally, section 2.6 sums up the chapter and positions it in terms of earlier work and the perceived needs in the area.

2.2 What is a Metaphor?

The aim of this section is to survey multiple perspectives on the concept of metaphor and then to synthesise a working definition of the concept. This definition will at first be *independent* of user-interfaces, before being narrowed down to apply only to them.

Before examining definitions, it will be useful to select a specific example to help explain concepts with. For this purpose, we will use Shakespeare's famous metaphor, identifying Juliet with the sun:

> But, soft! what light through yonder window breaks?
> It is the east, and Juliet is the sun!
> Arise, fair sun, and kill the envious moon
> [74, Act II, Scene II, l.1-3]

Aristotle's definition of metaphor is still very applicable today. He wrote that "metaphor consists in giving the thing a name that belongs to something else." [8, 1457b l.6-7] This characterises the basic nature of metaphor, in that it involves the identification of one thing with another. We will now go on to analyse this process in much more detail.

In the New Shorter Oxford English Dictionary is a two-part definition of metaphor. One part of the definition is that:

> [Metaphor is a] thing considered as representative of some other (usually abstract) thing [12]

This definition shows that that metaphor is considered a matter of *representation*. The concept of representation is itself a highly complex one, but the point here is that a metaphor uses one thing to stand for another. Thus, in the example, Juliet is identified with, or represented by, the sun. This feature of metaphor has various implications, some of which are apparent when considering the second part of the dictionary definition:

> [Metaphor is a] figure of speech in which a name or descriptive word or phrase is transferred to an object or action different from, but analogous to, that to which it is literally applicable [12]

This identifies two important qualities of metaphor: *difference* and *analogy*.

The quality of difference is important because if there were no difference involved, the metaphor would just be a statement of identity. If Juliet actually *were* the sun, then saying so would not provide any new information. In *Interface Culture*, Steven Johnson states that the property of difference is crucial to metaphor's success: "what makes metaphor powerful is the gap between the two poles of the equation. Metaphors create relationships between things that are not directly equivalent." [39, p.59] The magic of metaphor comes when two disparate concepts are related and some new meaning or explanation is created.

This quality of difference is also what leads Donald Norman to say that "a metaphor is always wrong, by definition." [64, p.180]. The use of the word "is" in metaphor will always be technically incorrect because the two things are *not* the same. It is the *strength* of that "is," however, that makes metaphor such a powerful device. The related literary device of simile compares concepts using "is like" rather than "is." It is a form of comparison, as metaphor is, but of a weaker kind. It explicitly acknowledges that the two things are *not* the same. Metaphor is interesting because it does not make this acknowledgement and so the relationship between the concepts is assumed to be that much stronger.

The process of transferring a name from one thing to another is important because it implies they have similar *properties* or *structures*. Metaphor involves not only the transfer of a name, but the transfer of properties and structure also. Romeo does not just call Juliet the sun as a *name*, he means that she has some properties related to the sun such as warmth, radiance and the giving of life. This is the quality of *analogy* mentioned in the dictionary definition above. Note that, if Romeo were to simply list those qualities by saying "Juliet is radiant, and warm, and life-giving" the statement is not only longer, but also intuitively less powerful. The metaphor suggests what cannot necessarily be captured in a simple listing of properties.

While the essential difference is what distinguishes the two concepts, the analogy draws them together and allows thinking of one in terms of the other. In this way, interesting unions can be formed which allow us to understand concepts in unique and often useful ways.

The Oxford Companion to Philosophy provides a very useful insight into the use of metaphor. It is stated that metaphors "are interpreted and they are interpreted differently by different readers and hearers." [37]. There is no guarantee that another person would attach the same qualities to Juliet as those listed above. Nonetheless, the metaphor very clearly conveys Romeo's admiration for Juliet.

The interpretive nature of metaphor only becomes an issue when the context of the interpreter is significantly different to that of the inventor or writer of the metaphor. The question of what context a person brings to their interpretation of a metaphor is an interesting one and is hinted at in Gerard Steen's literary definition. This is that "[metaphor is a] relation between language as an abstract system, individual language users, and cultural knowledge" [76]. Two main points can be taken from this definition. One is the importance of *culture* in metaphor: not only is metaphor interpreted by individuals, as discussed previously, but it is

interpreted in the context of culture. "Culture" could mean anything from broad country-based cultures, to religion, to specific social groups, and so on. Thus, people with different cultural backgrounds may interpret metaphors in different ways. A traditional European might understand "Juliet is the sun" much as Romeo did. Someone from a culture which actively *worships* the sun, however, might well take away a substantially more powerful, or possibly blasphemous, message.

The second important point to take from Steen's definition is that metaphor functions as part of a *system*. A metaphor is not isolated, but is *part* of the system of knowledge. When Romeo says that Juliet is the sun, this points to other concepts such as the sky, the moon, fire, and so forth. Concepts do not exist in isolation. Thus, when considering metaphor, it is necessary to look to the system of thinking it functions within. This is put well by Julie and Kenneth Kendall, in the context of Information Systems, when they say that "metaphors entail other metaphors in a web or network of associations, making them not stand-alone ideas" [41, p.38].

Steven Pinker's popular book on human cognition, *How the Mind Works*, offers an evolutionary perspective on why metaphor might have become useful to humans. Pinker claims that "we pry our faculties loose from the domains they were designed to work in, and use their machinery to make sense of new domains that abstractly resemble the old ones." [69, p.359]. What is meant by this is that humans interpret the world using sensory devices which were evolved for use a very long time ago. In those days, everything could more or less be taken at at face value. Pinker's suggestion is that in the new world of complex and abstract concepts, which are often not directly perceivable, we need new ways to help us understand things. One way of doing this, he says, is to use metaphors which link these new concepts to old concepts we do understand with the physical senses that evolution has given us.

Note also the mention of *abstraction* in Pinker's statement. The resemblance of the explaining concept of a metaphor to the concept to be explained is often an abstract one, apparent only through human thought. For example, the literal similarities between Juliet and the sun are few if any, and yet the identification or the two *does* make sense to us. Metaphor works at a different comparative level to the purely literal. The comparison exists between qualities and structures, rather than physical facts.

Another interesting perspective on metaphor concerns its importance to human thought. George Lakoff and Mark Johnson state that "our ordinary conceptual system, in terms of which we both think and act, is fundamentally metaphorical in nature." [43, p.3]. Again there is the idea of metaphor functioning within a *system* of thought rather than in isolation. More importantly, perhaps, if Lakoff and Johnson, and also Pinker above, are correct, metaphor is a key element of human cognition. In this case, it would seem that metaphor is crucial for helping humans comprehend new concepts. More will be said about Lakoff and Johnson's theory of metaphor in chapter 3.

Before suggesting a definition of metaphor it is useful to remember that metaphor can take different forms. Importantly, metaphor is not a purely linguistic phenomenon. Visual metaphors are frequently encountered in everyday life. In a television advertisement for toothpaste a little green monster might feature as a metaphoric representation of plaque, for example. Accordingly, it is useful to think of metaphor as something that can be realised in multiple forms, from language to visual images and more.

At this point it is possible to synthesise the above discussion of metaphor into a working definition. First, a basic characterisation of metaphor will be stated, followed by some important additional qualities.

> **Metaphor**: A device for explaining some concept or thing by asserting its similarity to another concept or thing.

Additionally:

1. Metaphor is, by definition, inexact.
2. Metaphor is interpreted contextually at many levels, from single individuals to large-scale culture, and everything in-between.
3. Metaphor functions within a system of thought, not in isolation.
4. Metaphor may be one of the key methods humans have for understanding new concepts.

2.3 Metaphor and Computing

Computers are one of the most complex and hard to understand man-made artifacts. What is more, a large proportion of people must use them on a day-to-day basis. There is a conflict here: many people, with varying abilities, all must understand how to use an extremely complex device to achieve results important to them. Finding ways for people to more easily use computers has become one of the key pursuits of those interested in computing. The field that specifically investigates the boundary between humans and computers is typically called Human-Computer Interaction.

One of the very common ways that people seek to intuitively understand computers is through metaphor. In the previous section it was shown how metaphor uses one concept to help explain or understand another. More specifically, it can use something *already* well understood to explain something not well comprehended. Because of this ability to explain new concepts in terms of old ones metaphor is used at all levels of interaction with computers.

At the hardware level, for example, the term "memory" refers to particular pieces of hardware inside the computer. "Memory" is a metaphor for what this hardware does: it stores data, or, in metaphoric terms, it remembers information. Many high-level programming languages use the metaphor of "objects" to understand program constructs better. Assembly language uses the metaphorical term "jump" to convey the meaning of a certain instruction to the computer. In the user-interface metaphor is used constantly. From the "trash-can" to the "document", metaphors enable users to understand how the underlying system is working. Metaphor is used with great frequency to help clarify some of the complexity engendered in computers. In fact, it is hard to imagine using computers at all without the explanatory powers of metaphor.

Computers are not the only devices made more accessible through metaphor. Metaphors are used to help coping with the complexities of any new technology. In his book, *Interface Culture*, Steven Johnson examines how this is done. He states that "every age comes to terms with the latest technology by drawing upon imagery of older and more familiar things." [39, p.16] In particular, metaphor is applied frequently to technology to make people more comfortable talking about new things. For example, consider the use of the term "bonnet" to describe the sheet of metal which covers a car's engine. This is an instance of the metaphor THE SHEET OF METAL OVER THE ENGINE IS A BONNET, and it makes that aspect of the car's functioning more obviously accessible to the people who use it.

Evolution has not equipped humans to directly comprehend any of the new concepts which enter our lives, let alone highly complex devices such as computer. Instead, "the transition from old technology to new is eased by metaphor" [78, p.135]. When new technologies arrive new metaphors are found to apply to them. The internet spawned metaphors such as "surfing" (see [39, pp.107-108] for some analysis of this) and "mapping" [26]. Indeed, Mark Stefik's book *Internet Dreams* is devoted to the description and analysis of metaphors involved in our comprehension of the internet [77].

As seen above, there are different levels at which one can understand computers. People who must deal with each of these levels tend to use metaphor in some way to make the complexity more bearable, from chip-designers through to ordinary users. The user-interface, however, is probably the richest source of metaphors in computers. This is because the average user of a computer is the least likely of any kind of computer-user to have an understanding of how their computer really functions. In fact, user understanding may be based almost entirely on metaphors.

2.4 What is a User-Interface Metaphor?

In the context of user-interfaces, metaphor is used in a slightly different way to the literary sense of "Juliet is the sun". This section will roughly outline the more specific concept of the *user-interface metaphor*. The focus of this thesis is firmly on the graphical user-interface and the use of metaphor within it.

It is relatively difficult to find a solid definition of a user-interface metaphor in the literature that is independent of evaluative concerns, or that is even stated *as* a definition. In fact, it appears it is generally taken for granted that people know what the concept means because they have encountered *instances* of user-interface metaphor. This has likely contributed to the sometimes confused nature of discussion of the concept. For this reason a reasonably strict definition of the user-interface metaphor will be put forward here, and then analysed much more deeply in chapters 3 and 4.

The approach to metaphor in general from section 2.2 can be examined to see how some of those concepts might apply to the user-interface. Clearly, the idea of *representation* introduced earlier will play a crucial role in user-interface metaphor because the user-interface is entirely a matter of representation. That is not to say that it is entirely *metaphorical*, but the images on the screen are all *representations* in the most basic sense.

Metaphor is a device for explaining one concept by identifying it with another. It is safe to say that, with user-interface metaphors, what is being explained is the underlying functionality and characteristics of the software which the interface gives access to. In this section the term "system" is used to refer to the target of metaphorical explanation in the user-interface.

Metaphor in the user-interface can be represented through multiple senses, rather than simply through language use. Traditional user-interface metaphors, such as the "desktop" involve images, and sometimes sound, as with the "ringing" sound in Microsoft NetMeeting. There is, in addition, the possibility of using haptic interfaces [6, 72] which would allow tactile representations of metaphor. The kinaesthetic sense is involved through the use of the mouse, for example, when "dragging" files from folder to folder. Although taste and smell are outside current standards, the fact that multiple senses can be engaged in the user-interface is likely to lead to more comprehensive and interesting use of metaphor.

The fact that metaphors are *interpreted* is extremely important in the consideration of user-interface metaphor. When users encounter a user-interface metaphor they bring their own context to the understanding and decoding of that metaphor. This context involves their culture, their line of work, their education, and all the qualities that identify them as an individual. Clearly, this is a major issue for the designer of a user-interface metaphor. It is inevitable that the target audience of the software will bring a *range* of contexts, which the metaphor will be interpreted through. This makes predicting how different users will react to a metaphor quite difficult. It is desirable, therefore, to define a common ground, and, as far as possible, avoid the confusion of highly varied interpretations.

It is also important that the explaining concept of the metaphor is familiar to the user. The purpose of metaphor in general, and user-interface metaphor in particular, is to explain

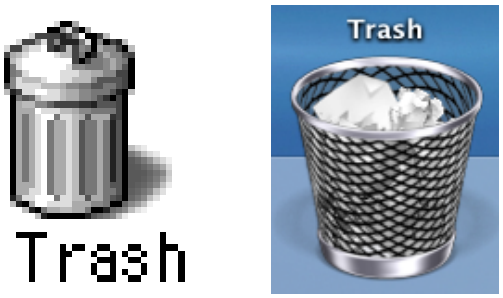


Figure 2.1: The trashcan from MacOS 9, and the version in MacOS X.

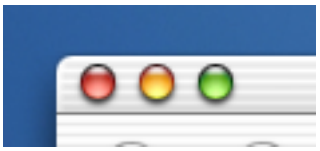


Figure 2.2: The traffic light window controls in MacOS X.

a concept not previously well understood in terms of something that is already understood. Alternatively, the metaphor might *enhance* our understanding of an already understood concept. The explanatory concept of a user-interface metaphor must be something familiar to the *user*, because it is the user who must ultimately understand and interact with the system.

Based on this discussion an initial definition of user-interface metaphor can now be offered:

> **User-Interface Metaphor**: A device for explaining some system functionality or structure by asserting its similarity to another concept or thing already familiar to the user.

It will now be instructive to examine some user-interface metaphors to test this definition. By doing this, the definition will be shown to capture the normal understanding of user-interface metaphor. The following examples will also serve to illustrate some of the subtleties of the concept.

Consider a familiar example of a user-interface metaphor: the "trashcan" (figure 2.1). If phrased in language, like "Juliet is the sun," it would perhaps be something like "file deletion is using a trashcan". Clearly, this is metaphoric because it uses something familiar to the user (a trash-can) to represent an aspect of the system that might otherwise be confusing (file deletion and management). Just as the sun explains certain features of Juliet, the trash-can explains certain features of file-deletion in the system. You can place unwanted items into the trash-can to get rid of them, and you can get them back if you are in time. Additionally, the trashcan is a *perceivable* aspect of the system, using the user's visual sense to convey the metaphor as well as the kinaesthetic sense of "putting" items into it.

As another example consider a more subtle metaphor: the window controls from MacOS X, as shown in figure 2.2. These controls represent some of the functionality of a window through the concept of a traffic light. The perceptual part of this interface element may not be *completely* faithful to the concept, but the round shapes of the buttons and their colours do evoke a traffic light. In the metaphor red means "close" (stop), amber means "minimise" (slow down or delay) and green means "maximise" (go). This raises the point that elements of the user-interface can be metaphorical in ways that are not immediately obvious. The fact that metaphors in the user-interface are not always easily recognised is an important area of exploration in this thesis and will be addressed in detail in chapters 3 and 4.

Probably the best known user-interface metaphor is the "desktop metaphor". This metaphor is present in the graphical interfaces of many modern operating systems. It is perceived through a collection of *sub*metaphors such as the trash-can, documents and the actual "desktop" surface. The desktop metaphor may be more appropriately called the "office metaphor", in fact, as the collection of sub-metaphors more accurately characterises an office working environment. The metaphor exists to help the user understand the function of their computer system as the working of an office, particularly as relates to data management. The user encounters "files" and "folders" as ways of representing the more complex data structures that the computer *really* uses. This raises the interesting point that metaphors can be applied at different *levels*. They can be more overarching, organisation metaphors, such as the desktop; more specific, like the trash-can metaphor; or even quite subtle and unnoticed, such as the traffic light metaphor.

It should be clear from the above examples that the definition of user-interface metaphor given captures the general range of user-interface metaphors. Chapters 3 and 4 go into greater depth in examining the concept of the user-interface metaphor. The next step here is to examine the positions taken on user-interface metaphors in the research community, where it has been studied in some detail. What will be shown, through this literature survey, is that there is some confusion about what user-interface metaphors can and cannot do, and very little principled understanding of the concept.

2.5 User-Interface Metaphor Research

User-interface metaphor has been widely adopted in industry. Both Apple and Microsoft have recommended the use of metaphor in their design guidelines:

> You can take advantage of people's knowledge of the world around them by using metaphors to convey concepts and features of your application. Use metaphors involving concrete, familiar ideas and make the metaphors plain, so that users have a set of expectations to apply to computer environments. [7]

> Familiar metaphors provide a direct and intuitive interface to user tasks. By allowing users to transfer their knowledge and experience, metaphors make it easier to predict and learn the behaviors of software-based representations. [48]

Sparking this wide usage, and as a response to it, has been considerable research into the concept of the user-interface metaphor by the academic community. The bulk of the research took place between the early eighties and the mid-nineties, closely tracking the rise of the graphical user interface. Recently there has been less academic focus on the concept, although there is still interest.

When considering research into user-interface metaphor it is useful to create some rough categories to make a survey of important material more comprehensible. First of all there is a division between theoretical and empirical studies. That there is *very little* hard empirical research available is a key point to note. Additionally, what little empirical research there has been is largely critical of the value of metaphor. Most of the research has been based on theory, with some researchers performing relatively informal studies of their ideas. This weighting toward theory is perhaps because interface metaphor can be difficult to quantify and thus hard to test with empirical studies. An additional reason for the apparent lack of rigorous empirical studies is that that there is a strong *assumption* that user-interface metaphors are a useful device. Many researchers appear to begin with the assumed usefulness of metaphor, and base their work around this [38, 45, 77]. This is often also reflected in the guidelines offered, such as those of Apple and Microsoft above.

Three further categories of research contrast work concerning benefits, work which criticises the idea of metaphor and, finally, work that seeks to present methods for actually creating or designing user-interface metaphors. These categories will each be examined in turn to provide an understanding of how user-interface metaphors are perceived and what research has already been performed. Section 2.6 will then position this thesis with respect to what has gone before.

This thesis will take a neutral stance on the benefits and limitations of user-interface metaphors. The goal here is to introduce a better way of *thinking about* user-interface metaphors that is independent of their merit.

2.5.1 Claimed Benefits

The essential benefit attributed to user-interface metaphors is that they increase or promote initial familiarity with the system. As we shall see, it is often suggested that this leads to a reduced learning period and higher user satisfaction. The basic argument is as follows: user-interface metaphors present a complex system concept in terms of a concept already understood by the user; the user is immediately comfortable with the interface, therefore, and already knows how to use it.

This may be the most optimistic version of the argument, but it is clearly what is in mind for many proponents of metaphor use.

John M. Carroll suggests that the metaphor approach "seeks to increase the initial familiarity of actions, procedures and concepts by making them similar to actions, procedures and concepts that are already known." [15, p.67] Thomas Erickson puts it similarly when he claims that metaphors "function as natural models, allowing us to take our knowledge of familiar, concrete objects and experiences and use it to give structure to more abstract concepts." [32, p.66]. Similar comments are made throughout the literature [25, 39].

A consequence of this proposed "pre-familiarity" with the interface, is that the user might be completely unaware of the underlying system. The suggestion is that, ideally, the user-interface metaphors are so convincing that the user is effectively unaware that they are using a computer system in the first place, or are at least unaware that the system is anything different from what they see. Rob Swigart puts this point of view across nicely in his article describing his daily interaction with a computer as a writer:

> I use the computer, and its metaphorical desktop has faded from my awareness, almost as if it were a genuine continuation of the horizontal surface of the desk itself, instead of some electronic window on a virtual world that ceases to exit when I turn the computer off. [78, p.141]

According to Swigart the metaphor becomes *the* way that the user think about the system. The underlying system becomes invisible: the user is no longer working with a computer representation of a desktop but, in some sense, just a desktop, as real as the desk the computer sits on. While this seems an optimistic view, it reflects the hope of many designers that a good metaphor will render the underlying complexity of the computer system invisible and unproblematic. In fact, some metaphors do become such a part of the user's thinking that they forget they are there. In an article on the evolution of the Xerox Star system, the writers, who were also part of the development team suggest:

> When Star displays a directory, it (unlike MS-DOS and Unix) is not displaying a list of the names of the files in the directory, it is displaying the files themselves so that the user can manipulate them. [38, p.57]

On a first reading this statement seem perfectly reasonable, but notice that the notion of a "file" here is, in fact, a user-interface metaphor. A computer file is a metaphor using the real-world object of a collection of pieces of information on paper. By saying that Star "displays the files themselves" the writers indicate that the file metaphor has become subconscious, as though the pictures of the files on the screen actually *were* the files. What is important, though, is that this seems *right* to us: those *are* the files. It will later be shown that many metaphors seem to be used unconsciously in this way.

Note that all the benefits claimed for metaphor are based on a certain level of informality. The benefits are posited through subjective experience, theory or, at best, informal testing of concepts. The general lack of formal empirical testing of user-interface metaphors may well be due to the difficulty of understanding user-interface metaphor as a concept in the first place, or possibly because metaphor simply seems natural to use.

2.5.2 Criticism

There are many critics of user-interface metaphors, offering various perspective on their problems. The desktop metaphor [44, 81], the trash-can metaphor[80], the paper metaphor in word processors [14], and metaphors for the Internet [39] have all been seriously criticised. Many researchers, however, have questioned whether the actual *concept* of the user-interface metaphor is valuable in the first place. In this section theoretical objections to the claimed benefits will be considered.

One major criticism concerns the key element of *difference* in metaphor, discussed in section 2.2. It is claimed that the imperfect mapping between the user-interface metaphor and the underlying computer system will cause difficulties for the user. Donald Norman sums up this line of argument in his book *The Invisible Computer*:

> Metaphors are an attempt to use one thing to represent another, when the other is not the same. But if it is not the same, how can the metaphor help? [64, p.180]

This claim is somewhat disingenuous, but does lead to a useful insight. It is in the nature of metaphor to have a distance or difference between the two concepts. This is understood implicitly by those who create metaphors, and those who interpret them. The claim, however, that the differences between the two concepts might *outweigh* the usefulness of a metaphor is quite potent. Capturing this concern, Norman suggests that "the metaphor can get in the way of learning" [64, p.180]. The traditional response to this from advocates has been to agree that this is an issue that needs to be carefully examined when using metaphor [15, 51].

A related problem, raised by John Carroll and Robert Mack, is that "not all aspects of all systems can be as easily and elegantly fitted to metaphors." [14, p.713]. It would be impossible to fit all details of a system into metaphors, so there must be aspects of the system which are *not* metaphorically represented. It is hard to say, however, that this is a flaw in the concept of the user-interface metaphor. It is rarely claimed that user-interface metaphors are useful only if they encompass and explain the *entire* system. That said, Jakob Nielsen seems to suggest as much when claiming "a metaphor may be considered as a model of the user's actual conceptual model of the computer system" [56, p.147]. Despite this, Nielsen, one of today's foremost usability experts, recommends moderation as concerns interface metaphors. Although he agrees it can be useful for unifying design, he also cautions against compromising usability for the sake of metaphors [60].

Instead, metaphor is typically suggested as *a* technique for making the system easier and more natural to use. For example, Thomas Erickson recommends metaphor only for problem areas of a system [32] and Lucy Wozny specifies that metaphor are for organising *elements* of the system [85]. Claims such as Theodor Nelson's that "once the metaphor is instituted,

every related function has to become a part of it." [54, p.237] are not actually suggested by those in favour of its use.

A second criticism raised by Nelson in his paper "The Right Way to Think About Software Design" concerns the nature of user-interface metaphors as being for *introducing* users to systems, rather than supporting them throughout their user lifetime. Nelson claims that users do not want to be "tied to any details of some introductory model." [54, p.237] Metaphors are often seen as a device for *new* users, and a serious critique is that they become a hindrance once the system is understood: they out-stay their welcome and get in the way. This issue is not easily dismissed and is also raised by others [14, 64]. Only a small amount of work has been done to examine the effect of user types on metaphor use in user-interfaces [16]. Overall, then, this is an area which seems to require significantly more research.

Another criticism related to the "difference" arguments above concerns a rejection of the claim that user-interface metaphors aid new users' learning process at all. Nelson suggests that they are not useful because "you must learn the non-obvious aspects of a lot of poorly designed screen furniture and visual toys: what they actually do, rather than what they suggest. You must explore the details of each until you understand what it "really means"." [54, p.236] Although this criticism clearly applies to user-interfaces in *general*, rather than specifically user-interface metaphors, it raises an important point. It is quite possible that designers using user-interface metaphor might lean too heavily on the assumed utility of the device and thus tend toward poor design. This is evidenced in poor designs where, for example, the book metaphor is taken to such an extent that users must *turn the pages* to progress through the software. Note, however, that this constitutes a warning against *badly* using metaphor, rather than metaphor user in general.

One final criticism comes from Alan Kay, who asks whether designers should "transfer the paper metaphor so perfectly that the screen is as hard as paper to erase and change?" [40, p.199] It is obvious that this is undesirable, and what Kay wishes to emphasise is that "it is the magical part that is all important and that must be most strongly attended to in the user interface design." [40, p.199] In other words, Kay feels that the important parts of the user-interface are the parts that the metaphor does *not* explain, such as how the user can change the style of text without completely re-writing it. It is important to remember, however, that metaphor is *by nature* partial. The metaphor "Juliet is the sun" is understood without, for example, thinking that Juliet is largely composed of hydrogen. Given that this gap between the concepts is an accepted feature of metaphor, it seems that it fits in well with what Kay desires in the interface. What can be taken from Kay is that designers must pay very close attention to the differences between the system and the explaining concept, as well as the similarities.

2.5.3 Empirical Criticism

John Carol and Robert Mack, in a study of a word processor program, found that its users repeatedly took the apparent typewriter metaphor *too* literally [14] . This relates to earlier criticism in an interesting manner. As was described above, Alan Kay criticised user-interface metaphor from the perspective of implementing too many of the implications, including undesirable ones. Carol and Mack's results suggest that users *do* expect many or all of the implications to be implemented. As they say, the users "were not able to resist referring to their prior knowledge about typewriters as a basis for interpreting and predicting experience with word processors." [14, p.709] Elsewhere Carol and Mack [15] reference a study in which 62 of 105 errors collected from text editor learners were attributable to mismatched metaphor mappings.

Nevertheless, it is worth remembering that as technologies become more familiar, the

tacit knowledge that users will bring increases. Although it has not been empirically shown, it is a reasonable assumption that today most users know not to expect exact typewriter-like behaviour from a word-processor. In fact, most of today's users will never have *used* a real typewriter. Despite this, The problem of mismatched expectations is still likely the largest issue for user-interface metaphor design [15, 16, 36, 75, 85].

A final empirical study to consider is Alan Blackwell's PhD thesis on the place of metaphor in diagrams [11]. Blackwell performed several different studies in seeking to establish how useful metaphor really is for explaining technical concepts. He was particularly interested in problem solving activities in visual programming languages, which involve much use of metaphor. His first study compared a literature survey of academic papers on visual programming languages with a survey by Blackwell of industry users of those languages. His results suggest that, while visual programming languages are very popular with researchers, the people who use them do not find them as useful as the researchers suggest they should. He concludes that the literature is over-enthusiastic about the concept of visual programming languages.

The rest of the thesis involves experiments with people using different visual languages, designed by Blackwell, to perform certain tasks. In these experiments systematically metaphorical notations are compared with what might be considered "nonsense metaphors." Blackwell's results suggest that there is little or no difference between the two techniques. Particularly interesting is Blackwell's finding that using pictorial representations in general for problem solving appeared more useful. Users could create their *own* metaphors, rather than necessarily follow the metaphor Blackwell had in mind. The results ultimately lead him to question whether metaphor really is as useful a tool as is generally claimed in the literature.

Although Blackwell's analysis of metaphor is useful, there are some aspects of his studies which are limited. In particular, the thesis deals with a very narrow aspect of metaphors in user-interfaces, that of visual programming languages. While it is an important area, it is not clear that Blackwell's conclusions will generalise to user-interface metaphors in other areas. Further research like Blackwell's into more common user-interface metaphors would be valuable in further determining the place of metaphor in interface design.

2.5.4 User-Interface Metaphor Design

While criticisms and claimed benefits for user-interface metaphors abound, attempts to explain how to *design* with them are less plentiful. There have certainly been attempts to formalise or inform the process, though, and some of the literature on these will be reviewed in this section.

In what is one of the most influential papers in the area, John Carroll, Robert Mack and Wendy Kellogg outline an approach for user-interface metaphor design [15]. The paper is recommended by Apple in their Human Interface Guidelines [7] as a key reference for using interface metaphors.

Their approach is to identify as many metaphors as possible that *could* be used, and then to apply them to the software design in order to establish their appropriateness. Appropriateness is determined by examining the "mismatches" that arise between the system and the explaining concept chosen. A mismatch is defined as when either the explaining concept suggests something not present in the system, or the system contains some function or structure not explained by the explaining concept. As noted earlier, these two factors are generally considered to be the greatest difficulties in using interface-metaphors.

Carroll et al. suggest two main methods for dealing with mismatches. One is to encourage *exploration* by the user so that they encounter the mismatches and learn how to deal with them. The second method is to use "composite metaphors": using more than one metaphor

in order to gain greater coverage of the system. Other benefits of composite metaphors are also cited, such as the ability to generate more inferences about the nature of the system, an integrated understanding of the system, and encouraging a more integrated and abstract view of the system.

The overall design process recommended by Carroll et al., then, is to generate possible metaphors by examining past software, the users themselves, and simply inventing them. Next, mismatches are identified, aided by examining user scenarios and listing the objects involved in the metaphor chosen. Advice is then given on how to manage the mismatches, as noted above. Overall, Carroll et al. offer a fairly comprehensive approach to design with metaphor.

In his oft-cited paper on the subject, Thomas Erickson offers an alternative approach [32]. His idea is to find a metaphor for each instance of problematic functionality in the system. For the process of evaluation he offers several categories in which to consider the chosen metaphor: the amount of *structure* offered by the metaphor; how *applicable* the metaphor is to the problem in question; how easily you can *represent* the metaphor in the interface so that the user will recognise it; how *suitable* the metaphor is for the users of the system; and how much you can *extend* the metaphor if needed. Erickson's approach to the overall design of the metaphors is to suggest examining the users' problem areas and then generating as many metaphors as possible that might help, evaluating them according to the criteria just discussed. His recommendations for this are to examine the problem description for implicit metaphors, and general invention.

Kim Halskov Madsen offers another a set of extensive general guidelines for developing and assessing the capability of user-interface metaphors [46]. The guidelines are effectively a large set of heuristics that concern the generation of metaphors, their evaluation, development and characteristics. The collection of guidelines is made up both of guidance from other papers, such as those of Erickson and Carroll et al., and original thoughts on the process. Some of the guidelines are based on five case-studies Madsen performed on different software types. While Madsen's guidelines are too numerous to reproduce here, there is certainly coverage of many of the issues involved with user-interface metaphors and some simple guidance as to how to circumvent them. Examples include guidelines such as "build on already existing metaphors", "choose a metaphor suitable to the audience" and "metaphors provide detailed and specific design options."

There are many other papers offering guidelines for the evaluation or design of user-interface metaphors in different situations and for different purposes. Jay Lundell and Steve Anderson offer some advice based on their experience on designing a piece of software for Unix in [45]. In [82], Kaisa Väänänen and Jens Schmidt consider some guidelines for creating good metaphors in the design of hypermedia. Cataci et al. address the use of metaphors for database systems in [17]. Another fairly detailed design process is offered by Smyth, Anderson and Alty in [75], which emphasises a set-based model of metaphors. In particular they identify four separate areas coinciding with the intersection of the system and explaining concept, the non-intersecting areas and the rest of the conceptual domain. Through this they feel it is possible to better analyse particular metaphors. Carroll and Thomas, in [16], provide a set of recommendations when using metaphors in the user-interface. Examples of their analysis include, "find and use appropriate metaphors in teaching the naïve user a computer system" and "consider the probable consequences to users and system designers of each metaphor used." Halasz and Moran suggest that metaphor or analogy is best left to explaining single concepts better, rather than whole areas of a system [36]. In [41], although not directly user-interface related, Kendall and Kendall offer advice on the user of metaphors in information systems development.

2.6 Conclusion

This chapter has moved from the general concept of metaphor to a detailed understanding of the concept as it relates to user-interfaces and the research around it. First, the meaning and importance of metaphor from various perspectives, including literature and evolution, was discussed. Next, the fact that metaphors are a crucial part of our understanding of computing was established. With this in mind, section 2.4 established a definition of the user-interface metaphor. Finally, a detailed literature survey of research in the area was performed.

It is revealing to examine some of the common threads in theories to date of how to apply user-interface metaphors. The guidelines offered have largely been common-sense approaches to metaphor use, sometimes based on informal case studies or user-testing. While this has certainly led to some very perceptive recommendations, the recommendations tend to be very general and there remains a definite place for a more principled study of the applications of metaphor in user-interfaces. Additionally, there is not a great deal of guidance on how to analyse the structure of metaphors, despite the frequent emphasis on this as a step in the design process.

Perhaps what is most lacking is a principled understanding of what metaphor actually *means* in the context of user-interface design. To that end, this thesis will present a taxonomy of user-interface metaphors which will much more concretely define and categorise the concept. With a good definition available, future work on the concept should be able to start from a firmer base, and thus achieve more specific results. It is hoped that a more rigorous and detailed definition will additionally make formal experiments more quantifiable and easier to carry out.

Chapter 3

Metaphors We Live By

3.1 Introduction

In their book *Metaphors We Live By* George Lakoff and Mark Johnson expound a substantial theory on the place of metaphor in everyday existence [43]. The book has been quite influential in human-computer interaction circles because of its strong emphasis on metaphor as a key to human cognition. As Lakoff and Johnson put it, through extensive linguistic evidence it can be shown that "our ordinary conceptual system, in terms of which we both think and act, is fundamentally metaphorical in nature." This is often seen as a justification for using metaphor in user-interfaces; the idea being that it will leverage a fundamental human ability.

The purpose of this thesis is not to assess whether metaphor is a *good* technique, but rather to enhance the understanding of the concept itself in the context of user-interfaces. Whether or not Lakoff and Johnson's claim that metaphors are fundamental to cognition is accurate, their analysis and classification of metaphor is extremely valuable.

In the process of emphasising the extensive usage of metaphor in every day human existence, Lakoff and Johnson create a highly taxonomic view of metaphor. In particular, they suggest that metaphors can be identified as belonging to particular classes, each of which has its own features and uses. By classifying metaphors in this way Lakoff and Johnson make it possible to analyse the nature of a given metaphor in considerable depth, particularly as regards its usefulness for achieving certain aims. The classification of metaphor also provides a more detailed vocabulary for their discussion in general.

This chapter examines the classifications of Lakoff and Johnson in considerable detail. In each section a new category will be discussed, first with an outline of Lakoff and Johnson's analysis, and followed by an application of the category to the user-interface. In this way, the links between Lakoff and Johnson's work and the user-interface can be made clear, and a thorough understanding of their research in general can be achieved. Section 3.2 explains orientational metaphors, section 3.3 deals with ontological metaphors and section 3.4 concludes the core analysis with structural metaphors. With these three key categories explained, section 3.5 goes on to examine the relationships between them all.

In the remaining sections, work from Lakoff and Johnson outside the three major metaphor categories is discussed. First, the concept of *metaphorical entailments* is introduced in section 3.6. This means of describing the meaning of metaphors is discussed in considerable detail in the following chapter and thus is addressed only briefly here. Following this, the concept of metaphor novelty is explained in some depth section 3.7. Finally, Lakoff and Johnson's work on the related notion of *metonymy* is expounded. Once again, each of the aspects will be discussed as it relates to Lakoff and Johnson first, and then with specific application to the user-interface.

3.2 Orientational Metaphors

3.2.1 Lakoff and Johnson

Orientational Metaphors are defined by Lakoff and Johnson as being metaphors which "give a concept a spatial orientation." [43, p.14] This spatialisation of concepts contributes certain ways of understanding, the chief of which is the ability to organise *systems* of concepts based on metaphors. In this chapter, the explaining concept of a metaphor will be called the *vehicle* and the concept to be explained will be called the *tenor*. Thus in the metaphor JULIET IS THE SUN, "Juliet" is the tenor, while "the sun" is the vehicle.

As an example of an orientational metaphor, consider the metaphor HAPPINESS IS UP.[1] In this case, the concept of happiness is identified with an upward orientation. Lakoff and Johnson posit the existence of the metaphor in every day life because of expressions such as "I'm feeling *up* today," "he has such *high* spirits," and "her spirits *rose*." The examples consistently identify happiness with an upward direction and provide evidence that the metaphor is used *systematically* by people.

One of the points that Lakoff and Johnson stress most in their book is that metaphors are strongly based in our physical experience of the world. That is why, they suggest, metaphors are such powerful and fundamental devices. Orientational metaphors are clearly a good example of this view: they directly identify concepts in need of explanation with the human spatial experience of the world. Lakoff and Johnson spend considerable time analysing what physical bases several metaphors have in their book. For HAPPINESS IS UP, for example, they suggest that "erect posture [goes with] a positive emotional state." [43, p.15] It is worth noting that Lakoff and Johnson claim their ideas for physical bases are only intended to be plausible, as opposed to definitive. Whether or not these physical bases are accurate or not, the concept of an orientational metaphor is what is interesting in this thesis.

The second key feature of orientational metaphors is that their focus is largely on concept *systematicity* rather than *explanation*. As Lakoff and Johnson put it, orientational metaphors can organise "a whole system of concepts with respect to one another." [43, p.14] Thus, orientational metaphors involve the *organisation* of groups of concepts, rather than necessarily revealing important facts about the concepts directly through the vehicle. Any revelation through the vehicle is done through the physical bases discussed above. The upward orientation is revealing about happiness in an *explanatory* sense only so far as erect posture and so forth helps to illuminate the concept of happiness. The *systematicity* of orientational metaphors can be divided into *internal* and *external* types.

The internal systematicity of orientational metaphors concerns the fact that a given spatial metaphor defines a coherent system of instances, rather than isolated examples. Thus, "I feel *up*" and "my spirits *rose*" refer to happiness, and are clearly coherent with each other. This coherence would not be present if both "I feel *up*" and "my spirits *fell*" referred to happiness. This coherence provides a strong bias toward always using this upward orientation, because to break from it will seem awkward and strange. Thus the internal systematicity reinforces itself.

Perhaps more interestingly, orientational metaphors are also *externally* systematic. What this means is that they systematise *groups* of concepts. Given that there are a limited number of spatial metaphors possible, such as up, down, left, right, and so forth, it is clear that many concepts will share the same orientation in their orientational metaphors. Because of this, the concepts sharing an orientation are implied to be related in some way through spatialisation. Thus, orientational metaphors emphasises the relationships *between* concepts,

[1] This thesis will follow the convention of Lakoff and Johnson in presenting metaphors in a small-caps font [43].

as well as offering a systematic way of speaking or thinking about a particular concept.

As an example, consider that, while happiness is associated with up, other concepts are also identified with the upward orientation. Thus, MORE IS UP, HIGH STATUS IS UP, and GOOD IS UP. In this case, the common spatial orientation of the concepts is informative because it indicates important relationships. In particular, note that this system implies that happiness, more, and high status are all *good* in some sense, because they share the same orientation. As another observation, consider that the system implies that happiness and high status are similar and even equivalent in some sense. These implications arising from the orientational metaphors of Western culture are extremely revealing about the way in which life is viewed in that culture. Thus, high-status means happiness in some ways, and more is better, according to these metaphors.

Not only do orientational metaphors organise concepts according to a single orientation, there is also a system implied by opposing orientations. Corresponding to the HAPPINESS IS UP metaphor is the metaphor SADNESS IS DOWN. Thus, a vertical system of "mood" is established, and it can be seen how they relate to and oppose each other. This is why, in a previous example, saying "my spirits fell" to mean that "I am happy" rings false. A downward orientation is associated with sadness in the Western mind.

As hinted at in the above discussion, orientational metaphors tend to be culture *specific*. This is well demonstrated by George Lakoff's example of group of people from Western Mexico who understand spatial locations through a body metaphor [42, pp.313-315]. Thus, they use phrases such as "he is located at the mountain's head," rather than "he is on top of the mountain," using the metaphor THE TOP OF SOMETHING IS ITS HEAD. While this metaphor is *comprehensible* to other cultures, it does not seem an immediately natural way to discuss spatial locations. The metaphor is, to some degree, specific to that culture.

3.2.2 The User-Interface

The concept of an orientational metaphor can quite straightforwardly be applied to the user-interface. As will be shown in this section, orientational metaphors are a fundamental form of metaphor use in user-interfaces and occur with great frequency.

One particularly interesting point to be made is that orientational metaphors are quite likely often used in the user-interface without the designer's explicit awareness. This is chiefly because they are such a basic part of the way we address certain concepts that they are barely considered metaphors at all. Much as a person does not necessarily think they speak metaphorically in saying "I feel so *up* at the moment," nor will designers always notice when they use orientational metaphors in an interface. Given this unconscious usage, it is important to bring to light just how orientational metaphors affect the user-interface. The examination of their common uses should enable designers to become more aware of their presence, as well as to gain the ability to utilise them purposefully to achieve particular effects.

The computer screen has, for a long time, been treated as a form of space. It has progressed from a simple, two-dimensional space to what is often called a two-and-a-half dimensional space, where elements on the screen can overlap. Additionally, of course, three-dimensional effects can be achieved on a computer screen. With this spatialisation of the user-interface interface comes the ability to associate system concepts with directions and other spatial notions. The uses of orientational metaphors in user-interfaces can be broadly put into two categories: *navigation* and *quantification*.

Computer software often involves the process of navigating through information in order to achieve the user's goals. Consider the metaphor NEXT IS TO THE RIGHT which is frequently used when navigating through a series of dialog boxes, such as in a "wizard." Very often, as

Figure 3.1: The "next" and "back" buttons from a wizard dialog box.

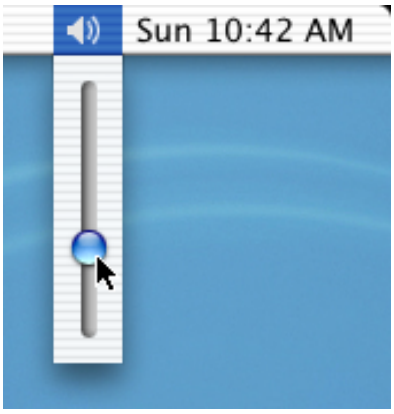


Figure 3.2: The vertical slider used to control volume in MacOS X.

shown in figure 3.1, the button to proceed to the next screen in a wizard will have an arrow on it which *points* to the right, while the back button has an arrow pointing left (BACK IS TO THE LEFT). These arrows directly associate the system concepts of progression and regression through the series of dialog boxes with the orientations of right and left respectively. This metaphor likely holds its basis in the way humans read text from left to right in Western cultures, and also ties in to the notion of the "number-line" which proceeds from left to right, as does a "time-line." It is quite likely that most designers would not consider the next and back buttons in a wizard as metaphoric, but in fact they are. Note that even this is not necessarily the whole story. In fact, the use of "back" indicates that dialog boxes are something to be moved *forward* and *backward* through. Thus, those two orientations have been re-mapped to left and right to fit a two-dimensional interface. This differs from most game situations in which a character is moved forward by pressing the *up* arrow. The mapping of directions between different dimensionalities and the corresponding manipulation of metaphors is an interesting design consideration. Over time, however, various implicit standards have arisen, and these seem to be usable.

Quantification is an important concept in working with computers. The specification of quantities is a frequent event in user-interfaces and many of the ways this specification is performed are metaphorical in nature. Consider, for example, the slider, a very common user-interface element. Vertical sliders traditionally work through the metaphor MORE IS UP. When the slider is moved upward, this indicates a desire for a greater amount of the quantity the slider controls. The physical basis for this metaphor suggested by Lakoff and Johnson is that "if you add more of a substance or of physical objects to a container or pile, the level goes up." [43, p.16] A typical example of a slider is the volume control for programs which produce sound output as shown in figure 3.2. In most cases the slider is moved upward (for vertical sliders) or to the right (for horizontal sliders) in order to indicate *more* volume is required. The association of "right" with more is likely due, once again, to the re-mapping of orientations such that, in one dimension, "up" or "more" become the right. This is reflected, for example, in number-lines and in graphs generally. Sliders are an excellent example of how a metaphor can be present in an interface element without any overt indications. It

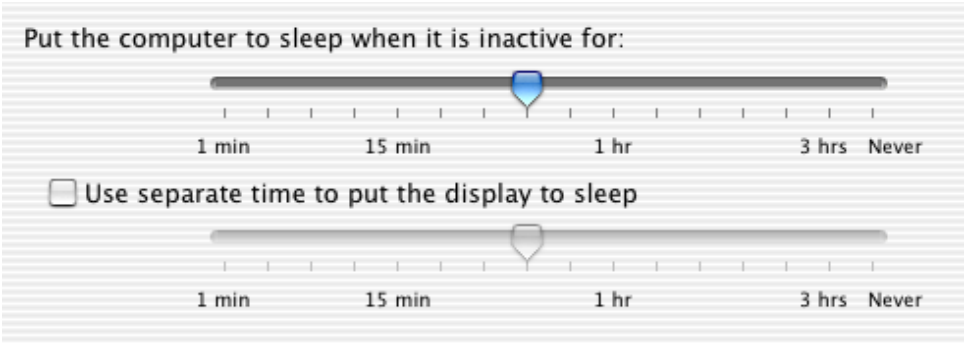


Figure 3.3: The horizontal sliders used to set computer sleep time in MacOS X.

is this subtle nature of orientational metaphors that makes their identification important. Additionally, horizontal sliders, such as those in figure 3.3 are used to quantify *time*, where more time is to the right.

The spatialisation concepts in a user-interface are an extremely important factor in determining its usability and, as such, should always be considered. In particular, note that by giving some interface element a spatialisation metaphor, the designer is *associating* that interface element with any other concepts in the user's mind which use the same orientation. It is important, therefore, to make sure, so far as is possible, that the orientations in an interface make the right kinds of associations. It would be a problem, for example, to use a slider in which a positive opinion of a presidential candidate was indicated by moving the slider toward the bottom. This would conflict with the system of metaphors which associate various *negative* concepts with a downward orientation such as BAD IS DOWN and DISSATISFACTION IS DOWN.

As a final note, it is quite likely that designers will tend to make the correct decisions anyway. This is because they will tend to share their orientational metaphors with their target users. Of course, this may not be the case when dealing with software design intended for other cultures, or designed to be cross-cultural. In these situations, considerable care must be taken and it is advisable to research what orientational metaphors are used in cultures being designed for before proceeding. Aside from cultural issues, it is equally important to insure that the orientational metaphors are consistent *within* an interface to avoid confusion. Although orientational metaphors can seem minor and difficult to perceive, they are bound to have a significant effect on the user if implemented incoherently or incorrectly.

3.3 Ontological Metaphors

3.3.1 Lakoff and Johnson

Ontological metaphors arise, Lakoff and Johnson say, because "our experience of physical objects and substances provides a further basis for understanding." [43, p.25] An ontological metaphor uses the stuff of every day existence, such as objects, entities and containers, to help explain other concepts. Because they provide links with such fundamental aspects of human interaction with the world, ontological metaphors are very basic and powerful. This is particularly due to their universal applicability. All humans know, at some level, what the properties of an object or entity are, for example. In other words, because humans all encounter the *same* basic categories of existence, they all have similar concepts about how these categories behave. Because of this, ontological metaphors are more likely to generalise between cultures.

As an example of an ontological metaphor, consider the case of INFLATION IS AN ENTITY. In this metaphor the concept of inflation is identified with the every day concept of an entity. Lakoff and Johnson find evidence for this metaphor in much human speech, such as "inflation *is hurting* our standard of living," "we need to *combat* inflation," and "inflation *is backing us* into a corner." All of these phrases address inflation as if it were an actual entity possible to interact with in a physical way and capable of causing events in the world. The metaphor provides a concrete way of discussing the concept of inflation. Note that Lakoff and Johnson also claim that this is how people *think* about inflation at some level: people actually consider it to *be* a kind of entity in the world, rather than only a purely abstract concept. In fact, it is hard to imagine how it would be possible to discuss inflation easily *without* resorting to an ontological metaphor. None of the commons ways in which people talk about the concept would be applicable and very long and obtuse phrases would have to be used instead. In fact, it is arguable that it is simply not possible to discuss abstract concepts without using some metaphor tying the concept to the real world.

There are *levels* of ontological metaphors: it is possible to move from the base ontological metaphor INFLATION IS AN ENTITY to the more specific INFLATION IS A PERSON, and finally to INFLATION IS AN ADVERSARY. Each of these metaphors builds on the previous one and adds certain new details which can be used to enhance understanding of inflation. In fact, it is arguable that the final metaphor in this example is of the structural type to be discussed in the next section. This blurring between the two categories will be examined in detail in section 3.5.

A particular kind of ontological metaphor that Lakoff and Johnson emphasise is the *container* metaphor. This type of metaphor is very popular, they claim, because humans "are physical beings, bounded and set off from the rest of the world by the surface of our skins" [43, p.29]. Lakoff and Johnson develop a complex account of how container metaphors are used in every day life, particularly emphasising the metaphor that EVENTS ARE CONTAINERS, leading to utterances such as "he is *in* the race" and "the game was *full of* excitement."

Lakoff and Johnson claim that ontological metaphors serve many different purposes. In particular, they suggest that they are used for *referring*, *quantifying*, *identifying aspects*, *identifying causes*, and *setting goals and motivating actions* [43, p.27]. By identifying concepts with the basic stuff of existence, such as objects and entities, humans are able to more easily perform these mental processes. As an example, note that an object metaphor makes quantification possible because physical objects are exactly the sort of thing humans are used to quantifying. This leads to the ability to quantify abstract concepts such as when someone says, "I have far *too much* trepidation." Similarly, the identification of causes is made possible only by recasting discussion in terms of the real world, where physical causation actually takes place and thus, "my fear *made me* run away."

It is important to realise that ontological metaphors are much more likely to generalise across cultures, as all humans encounter the same physical realities of existence. All humans understand the existence of objects, entities, containers and so forth. Although it might not always be easy to understand *why* a particular ontological metaphor is used in a culture, the general characteristics of these aspects of the world are common to all cultures. All objects have a solid form, they all have a weight, and they all occupy space. These things are constants, and thus an object metaphor will likely evoke those properties in whatever culture it is used. This commonality will make it more easy to understand the meaning of the metaphor although it may not aid in understanding it's *purpose* immediately.

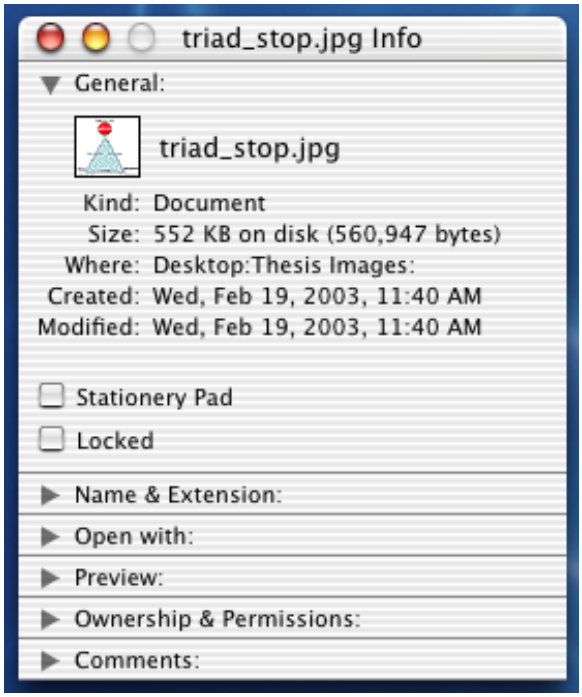


Figure 3.4: An example of an ontological metaphor used to quantify and define location in the user-interface.

3.3.2 The User-Interface

Ontological metaphors are used with great frequency in user-interfaces because they are so common to general human understanding of the world. Similarly to orientational metaphors, ontological metaphors are quite capable of being used without any genuine intention. This is once again because the metaphors deal in quite abstract, and not immediately recognisable, concepts. Given that it is not possible to create a direct representation of a vehicle such as "container" or "entity," it is not easy to directly perceive ontological metaphors in an interface. Instead, it is more sensible to identify them based on the *purpose* or *function* they serve within the interface.

As discussed above, ontological metaphors serve many purposes. Three of the specific purposes identified by Lakoff and Johnson are particularly relevant to the user-interface. By examining each of these uses of ontological metaphors, it is possible to gain a much better understanding of their function, as well as ways of identifying them.

The ability to use ontological metaphors to *refer* to things in a user-interface is one of their most important uses. By identifying system concepts with objects and entities, for example, it becomes possible to indicate their existence to the user. As an example, consider the document metaphor in modern computer systems where a collection of data is referred to as a document. Users might be instructed to "*move* the document into the trash" or "*throw away* the document." By identifying the system concept as an object, it becomes considerably more straightforward for the designer to indicate its presence in the system. While a concept by itself has no actual presence or form, an object has, by definition, both of these things.

Quantification is crucial to interacting with computers as already discussed with reference to orientational metaphors. Ontological metaphors provide a way to make quantification in user-interfaces more understandable. In particular, objects and substances are both things that are commonly quantified in every day existence. By using metaphors which identify system concepts with these things, this power of quantification is transferred to the interface

Figure 3.5: An example of an entity metaphor used to indicate causation in the user-interface.

domain. Hence, because A FILE IS AN OBJECT it is possible to talk about the file having a *size* as shown in figure 3.4. The object ontological metaphor additionally allows the interface to refer to the files *location*. In this way sets of data become less an abstract, amorphous concepts and more something "real" the user can quantify.

An extremely important task in user-interface is communicating to the user what is happening in a *causal* sense. The user requires some understanding of what causal processes are occurring in order to gain an understanding of the workings of the computer system. As an example of the prevalence of this concept in human-computer interaction, consider Ben Schneiderman's golden rule of "closure" which concerns making the process of an interaction apparent to the user by giving it a beginning, middle and end [73]. Ontological metaphors are used for this because system concepts can be identified with real world objects and entities which, in turn, are the *causes* of real world events. To illustrate this, consider the metaphor A PROGRAM IS AN ENTITY. In the user-interface programs are explicitly referred to as entities, as well as being implicitly thought of in that way. Thus, as in figure 3.5, a dialog notifies the user that the program needs to add some information to a folder. This message suggests that the program is an entity which needs to perform an action on the user's behalf. By thinking of the program in this way, the user can more immediately understand the causal processes within the user-interface.

Errors in the system are sometimes cast as objects (AN ERROR IS AN OBJECT) such as when a user says "the program has an error in it." In this case it is the basic physical quality of an object that is being used in the metaphor, rather than the causal ability of an entity. This fits real world examples such as an engine not working because "it has a stone in it," for example. Objects can passively cause events, while entities can actively cause them. The chief idea here, then, is to use an entity metaphor when talking about *intentional* causation, and an object metaphor when talking about *passive* causation. By identifying system concepts with objects and entities, it becomes possible for users to think in an ordinary, physical way about causation.

Container metaphors are also frequent in user interfaces. Consider the metaphor A TEXT-BOX IS A CONTAINER. A text-box in an interface is a place where the user can put text and which stores that text. This behaviour is the same as a container in every day life. By using this behaviour in an interface, an aspect of the world is harnessed to relate the interface back to physical existence, and thus one can "type some text *into* the text-box." Indeed, the name "text-box" itself clearly indicates that interface element's use of the container metaphor.

As has been shown, ontological metaphors are commonplace in user-interfaces. They are used for various purposes very common to computer systems. In fact, it seems they are

unavoidable if it is to be possible to deal with certain aspects of a computer system. Because of this, it is extremely important for interface designers to understand exactly where and how ontological metaphors are being used. This awareness will lead to a design more consistent with user expectations. Although ontological metaphors are quite natural and will thus tend to be used not only unknowingly, but *correctly*, it is well worth while being aware of their presence. In this way it will be possible to gain a better understanding of how the system is viewed in terms of its relation to physical reality. This, in turn, will aid an understanding of how users are likely to *reason* about the interface.

3.4 Structural Metaphors

3.4.1 Lakoff and Johnson

The most popular and familiar type of metaphor which Lakoff and Johnson identify in their book is the structural metaphor.

A structural metaphor involves a vehicle which is some well understood object or concept from the real-world. Structural metaphors differ from ontological metaphors in that they are far more specific in the vehicle chosen. Thus, rather than the concept of an object, a structural metaphor might utilise the idea of a garden rake as its vehicle. The more specific the vehicle chosen, the more structure it is likely to posses, and hence the name "structural metaphor."

As an example of a structural metaphor, consider Lakoff and Johnson's example of ARGUMENT IS WAR. This metaphor uses the structure of our concept of war to explain and illuminate the concept of argument. Lakoff and Johnson cite linguistic evidence for this metaphor such as "I felt *embattled* in the argument," "he *attacked* her position," and "she *shot down* his arguments." It is clear from these examples that people talk about argument *in terms of* war. Lakoff and Johnson move from this to the claim that humans *think about* argument with the structure of war, and *experience* it as a kind of war. Thus, war has become a fundamental means for humans to communicate and think about argument.

Structural metaphors are the most obvious kind of metaphor used in every day life. In particular, they are generally the kind used when metaphor is being used consciously. Shakespeare's metaphor JULIET IS THE SUN is an example of a structural metaphor. Structural metaphors are generally more apparent because the vehicle is a real item from the world which is likely to be familiar. Because of this, the disparity between the tenor and the vehicle will be more obvious and thus the metaphor is more likely to be detected. Consider the difference between "inflation is on its way" (the ontological metaphor INFLATION IS AN ENTITY) and "he can be such a pig sometimes" (the structural metaphor THE RUDE PERSON IS A PIG). The latter is clearly more easily identifiable as a use of metaphorical language.

3.4.2 The User-Interface

In general, when human-computer interaction researchers refer to user-interface metaphors, they are talking about *structural* metaphors. The traditional user-interface metaphor involves taking some real-world object or concept and using its structure to help the user understand a more abstract system concept. The concepts and objects chosen by interface designers tend to be recognisable every day things such as desktops and sheets of paper, rather than ontological or orientational concepts.

Examples of structural metaphors in user-interfaces abound, and any user-interface metaphor that is easily identifiable will generally be a structural one. Consider the "trashcan" as in figure 3.6. This embodies the structural metaphor FILE DELETION IS USING A TRASHCAN.

Figure 3.6: An example of a structural metaphor: the trashcan in MacOS X.

What it means to structure file deletion in terms of a trashcan is this: some of the properties and actions one can take on a trashcan are applied to file deletion in the file system. Thus, the user is able to perform actions such as throwing a file into the trashcan, or emptying the trashcan. Additionally, the metaphorical trashcan can be "full" or "empty." Most important, perhaps, is that it is possible to take something back out of a trashcan if you realise you still need it. In this way, facts about a trashcan, effectively the concept's structure, are transferred to the system concept of file deletion.

Above, it was noted that *some* of the properties and actions are transferred through a structural metaphor. As Lakoff and Johnson note, metaphoric structuring is, by nature, partial. This was discussed in the previous chapter also, where it was emphasised that metaphors are always incomplete. In particular, there will be facts about the vehicle that *do not* apply to the tenor, and there will be facts about the tenor that are not explained by the vehicle.

Because structural metaphors are the most specific of the three metaphor types, they can also be the most susceptible to cultural influences. In particular, when choosing actual things from every day life to explain a tenor, great care must be taken to account for the range of users expected to use the software. The more specific a metaphor is, the more likely it is to be a highly cultural reference, and thus not nearly as generalisable as something more abstract. This is emphasised by the fact that structural user-interface metaphors tend to be visually represented. This visual representation needs to be recognised by the target audience, and this restricts the acceptable choices. User-testing is a must when attempting to establish the appropriateness of a given structural metaphor. Thus, while using a vehicle such as "bidet" might be appropriate for software for use in France, it will not be as immediately recognised or understood by an American user-base, for example.

3.5 Relationships Between the Categories

It is interesting to note that there is a very strong relationship between the three categories of metaphor just discussed. In particular, it seems that they represent a spectrum of *abstraction* of vehicles. The levels of abstraction appear to increase from structural, to ontological, to orientational metaphors. Specifically, an orientational metaphor involves a vehicle which is a purely spatial concept. There is no way to represent this vehicle other than as an idea in the mind. Ontological metaphors, on the other hand, have vehicles which are an abstraction of real things in the world. In that sense, then, they are less abstract that orientational metaphors. Finally, structural metaphors are clearly the least abstract of the three, taking real objects and concepts from the world and using these as vehicles. While it could be

argued that the vehicle "object" is possibly just as abstract as the vehicle "left," there is no arguing that "fire-truck" is equally as abstract as "entity."

Perhaps the most interesting relationship between metaphor classes, as discussed in the previous section, is that structural metaphors appear to be ontological metaphors made more specific. In some sense, structural metaphors seem to involve shifting an ontological metaphor along a hypothetical axis of abstraction toward the less abstract. In fact, a structural metaphor can be considered as "filling in the details" of an ontological metaphor. Thus, a structural metaphor might specify the ontological metaphor X IS AN OBJECT by stating exactly what object X is. In terms of user-interfaces this is apparent when we compare the metaphors THE DATA IS AN OBJECT and THE DATA IS A DOCUMENT. The former is ontological, while the latter is structural. A document is a specific *kind* of object. It becomes clear, then, that structural metaphors, particularly those based on physical real-world objects, tend to *extend* ontological metaphors.

This is not always the case, however. Note, for example, that structural metaphors based on specific *concepts* rather than real-world objects do not extend ontological metaphors. As an example, consider the structural metaphor ARGUMENT IS WAR. There is no underlying ontological metaphor involved here because war is a concept rather than a physically embodied thing in the world. Thus, not *all* structural metaphors have an ontological underpinning.

Additionally, not every ontological metaphor must be attached to a structural metaphor when appearing in an interface. Consider the metaphor THE TEXT-BOX IS A CONTAINER mentioned earlier. This is an ontological metaphor in its own right, without a structural metaphor building on top of it. It could be argued that this suggests that there is no metaphor involved here at all, and that a text-box literally is a new kind of container. Although this is certainly a strong critique of metaphor, it is an extremely detailed and complex one, which would require considerable discussion. The debate centres around Donald Davidson's work the Philosophy of Language, and interested readers can turn there for more information [22].

A key issue with purely ontological user-interface metaphors concerns their identification. When they underly a structural metaphor, identification is fairly straightforward because it is possible to identify the structural metaphor relatively easily, and then to establish what underlying ontological metaphors it must have. Thus, having identified the structural metaphor THE DATA IS A DOCUMENT, it is straightforward to establish that THE DATA IS AN OBJECT simply because of the ontological categories that "document" belongs to. When ontological metaphors are present *without* an extending structural metaphor, however, they are much harder to identify. Purely ontological user-interface metaphors are not necessarily represented by things we recognise from the real world. A text-box, for example, is simply a white square on the screen. Because of this, identifying pure ontological metaphors relies on considering our interaction with them. When we realise that we put text in text-boxes to store it there, we can then note that the text-box is being treated as a container. The possibility of performing such identification was discussed earlier, along with the common functions of ontological metaphors in user-interfaces.

3.6 Metaphorical Entailments

3.6.1 Lakoff and Johnson

As a means to discuss the structure or content of metaphors, Lakoff and Johnson introduce the concept of *metaphorical entailments*. A metaphorical entailment is a statement of some aspect of the vehicle that applies to the tenor of a metaphor. Thus, for the metaphor ARGUMENT IS WAR, for example, some metaphorical entailments are that "positions can be attacked in an argument," "participants in an argument can feel embattled," and "arguments can be won or

lost, but sometimes it is hard to tell by whom." All of these statements are things related to war which are believed about or implied about argument through the metaphor ARGUMENT IS WAR. It can be claimed that the complete set of these metaphorical entailments about arguments effectively describe what the metaphor should be taken to *mean*. Lakoff and Johnson also stress that the structure of the entailments and their coherence are important factors in describing the content of a metaphor. Overall, then, metaphorical entailments can be considered as a semi-formal description of the meaning of a metaphor.

The most simple way to represent the process of developing a metaphorical entailment is as a deductive argument. This involves two premises: the metaphor itself, and some fact about the vehicle. Through these two premises it is possible to conclude with the metaphorical entailment:

1. ARGUMENT IS WAR
2. Positions can be attacked in a war
3. Therefore, positions can be attacked in an argument

Clearly, any fact about the vehicle could *potentially* lead to a metaphorical entailment. Equally clearly, not all implications can be valid. As already discussed, that would mean that the tenor and vehicle were identical. Thus, only a particular *set* of metaphorical entailments define a metaphor, rather than the complete set of all possible entailments. This is what makes metaphors difficult to quantify: different people may well attribute different sets of metaphorical entailments to the same metaphor. A concrete position will be taken on this in the following chapter. The key point is that metaphorical entailments can be treated as the content of a metaphor. This enables a more structured view of metaphors which is desirable in a design context.

3.6.2 The User-Interface

Metaphorical entailments apply straightforwardly to user-interface metaphors. Any given user-interface metaphor can be described by indicating the set of metaphorical entailments which define it. Usefully, this set of metaphorical entailments can be related in detail to the *functionality* underlying the user-interface metaphor. Only those entailments which are, in some sense, *implemented* are those which are applicable.

The concept of using metaphorical entailments to characterise user-interface metaphors will be discussed in considerable detail in the following chapter. The most important thing to know about them at present is that they are a means to more rigorously define a user-interface metaphor and to tie it firmly to the actual implementation of the user-interface. Metaphorical entailments provide a useful concept for the discussion of user-interface metaphors and will be shown to be an excellent tool for analysing them.

3.6.3 Coherence

An important consideration that metaphorical entailments raise is the notion of metaphorical *coherence*. Because metaphors have specific entailments, it is possible for the entailments of two metaphors to conflict with each other. Most importantly, coherence issues arise between two metaphors which are intended to explain the same concept. This occurs because metaphors can have overlapping metaphorical entailments, which make statements about the same thing. As Lakoff and Johnson put it, metaphorical entailments "play an essential role in linking *two different* metaphorical structurings of a single concept." [43, p.96] Thus, if two metaphors *share* a metaphorical entailment, this makes them coherent with each other to

that degree. Naturally, they will not share all metaphorical entailments, but the more they share, the more coherent they are with each other.

This possibility of having metaphorical entailments which explain similar things also leads to the problem of *incoherence*. An incoherence occurs when two metaphorical entailments from metaphors *conflict* with each other or disagree. In this case, the metaphors do not interact well with each other and will likely hinder understanding if they are used together. For example, consider the metaphors INFLATION IS A CAT and THE ECONOMY IS A MACHINE. Although these two metaphors may well work together in some ways, it seems inevitable that they will conflict considerably in their metaphorical entailments.

Naturally, the concept of coherence can be applied to the metaphors used in a user-interface. It is generally unlikely that metaphors in the user-interface will *specifically* cohere with each other by sharing some of the same entailments, but it is not impossible. Consider the example of the "document" and ''trashcan" metaphors which both involve entailments of "you can throw a document into the trashcan." Where coherence is available, it is a powerful concept, that reinforces interaction concepts to the user. The issue of *incoherence*, is likely more obvious, however. In this case, metaphors which are intended to explain system concepts conflict with each other in some way. There is some sort of incoherence involved in the ability to "lock" a document against editing, for example. A document in the real-world cannot literally be locked, to prevent someone from changing it, yet can be in the user interface. In this case the "lock" metaphor and the "document" metaphors seem to involve an incoherence. It is a decision for the designer as to whether this incoherence is serious enough to eliminate one of the entailments or possibly one of the metaphors.

3.7 New and Conventional Metaphors

3.7.1 Lakoff and Johnson

Lakoff and Johnson emphasise the important concept of metaphor novelty in their book. In particular, they suggest that any metaphor can be either *new* or *conventional*. New metaphors are those which "are outside our conventional conceptual systems, metaphors that are imaginative and creative." [43, p.139] Thus they are those metaphors which are not already well established in the general set of metaphors that people are accustomed to. Conventional metaphors, on the other hand, are those which are already understood by people.

Note that there is an issue of relativity here: different groups of people use different metaphors to express their concepts. Because of this, it is not possible to label any one metaphor as new or conventional without also referring to the *group* this is relative to. A metaphor might be new to one group, but conventional to another. As an example, consider the metaphor GOOD IS UP, so common to Western cultures. As Lakoff and Johnson point out, in other cultures this focus on up-down orientations is not as apparent and the focus is more on balance of centrality [43, p.24]. To such a culture, then, the metaphor GOOD IS UP would actually be new.

The other important quality of the novelty of metaphors is its ability to change. A metaphor might begin as new, but proceed over time to become conventional. Naturally, this still applies only relative to a particular group. Thus, at one time a metaphor such as THEORIES ARE BUILDINGS was very unique and strange. Over time, however, it has become completely conventionalised and is often no longer immediately recognised as metaphorical and phrases such as "building a theory," seem perfectly natural. This transition from newness to conventionality defines the life-cycle of a metaphor. In a sense, it also defines how successful a metaphor is: the more conventional a metaphor becomes, the more accepted is has been by

Figure 3.7: An example of assisting metaphor interpretation with secondary indications: the MacOS X window controls before and after a mouse-over.

the population. In a final case, perhaps, metaphorical uses become another *definition* of the vehicle. Thus, a using a word such a "bonnet" for the hood of a car, which was metaphorical, is now definitive.

The key distinction between new and conventional metaphors involves the interpretation of their underlying structure or meaning. A metaphor becomes conventional when those using it all largely agree on its metaphorical entailments. A new metaphor is one for which the set of metaphorical entailments is still being agreed upon. Conventional metaphors can be relied upon to convey an established meaning, then, while new metaphors might well need additional explanation to convey the intended set of metaphorical entailments.

3.7.2 The User-Interface

The new/conventional distinction is an important one in the design of user-interface metaphors. Specifically, knowledge of the distinction should lead to better design decisions as regards the user-interface.

Of particular importance is the relative nature of metaphor novelty, discussed above. It is critical for a designer to know who the target audience is to some degree of accuracy in order to be able to judge the newness of the metaphors they plan to use. This is, in turn, because the level of newness of user-interface metaphors dictates certain factors concerning how they should be utilised.

Before discussing exactly what these factors are, an example of each category will be provided. The user-interface metaphor THE DATA IS A DOCUMENT is largely a conventional metaphor. People who use computers generally know what to expect of a document on their computer system. They do not, for example, expect it to be made of paper, but they do expect to be able to put text in it. A new metaphor might be THE DATA IS A PAINTING with metaphorical entailments such as that data is created by people with a passion about it, and that data can be very valuable or worthless. While that metaphor has some relevance and utility, there can also be completely ridiculous new metaphors such as THE DATA IS A DOG. A user would likely not know what to expect when they encountered data that was interpreted through the explaining concept of a dog. There is no initial way of knowing whether the data would bark at them, run around the screen, or need to be walked and fed, for example. The dog metaphor also illustrates that a metaphor is not simply a matter of picking *any* two concepts, but only two that have some interesting and *useful* similarities.

The key issue, which is introduced above, concerns the standardisation of the metaphorical entailments of a user-interface metaphor. New metaphors must be very carefully designed to *show* the user which metaphorical entailments are applicable in the interface and which are not. Although some researchers recommend *exploration* as a means to understand user-interface metaphors, it seems this would be best combined with specific indications of use from the designer through the representation in the interface. Figure 3.7 shows the buttons on a window in MacOS X which are simply coloured jewel-like objects. The usage of these is the rather obscure metaphor of a traffic light, where red (stop) means "close," orange (slow) means "hide," and green (go) means "maximise." In order to aid understanding, the designers

of the buttons created symbols which appear when the mouse is moved over the area (shown in the second part of figure 3.7). These symbols are intended to assist in interpreting the metaphor, by giving a secondary indication of the function of the buttons.

Conventional metaphors must be handled carefully because the set of metaphorical entailments is generally well-known. This means that if the user-interface metaphor does *not* use all of the traditional entailments, this fact must be indicated by the designer through the representation to avoid user confusion. The most simple way of achieving this is by specifically not including aspects in the representation that would indicate particular functions. Thus, if a designer is using a VCR metaphor, but does not wish to have a 'pause' function, the representation should *not* include the traditional representation of a pause button. These principles apply both to transferring traditional language metaphors to user-interfaces, and to user-interface specific metaphors.

Because user-interfaces are a relatively young technology, it might be thought that conventional user-interface metaphors do not exist. In the time that the user-interface has existed, however, many conventional metaphors have been created. Examples include THE DATA IS A DOCUMENT, DELETING FILES IS USING A TRASHCAN, and THE WORD-PROCESSOR IS A TYPEWRITER. All of these metaphors are generally accepted by users of computers, and their specific sets of metaphorical entailments are understood quite well. Although each of the metaphors started as new, through continued usage, they have moved toward conventionality. It is presumably a mark of a successful user-interface metaphor that it reaches conventionality rather than being discarded.

One key point that emerges from this discussion of novelty is that user-testing is of major importance when examining user-interface metaphors. The only way to establish whether a metaphor should be considered new or conventional is by testing it with members of the target user population. While certain estimates can be made, it is by finding out what actual users think that a good judgment can be arrived at. A possible process would be to discuss the notion of metaphorical entailments with users and then have them list the metaphorical entailments they believe to apply to a given user-interface metaphor. In this way some sort of description of the user's impression of the metaphor could be developed. This list of the user's metaphorical entailments could then be compared with those conceived by the designers of the interface which would lead to an assessment of how well the user understood the designers' intentions. By considering the conflicts and matches between the two sets, useful information about the interface could be derived, particularly as regards problem areas in the design.

3.8 Metonymy

3.8.1 Lakoff and Johnson

The final aspect of Lakoff and Johnson's work that is of interest is the notion of *metonymy*. This is not a kind of metaphor, but is related enough that it belongs in this discussion. In particular, metonymy is similar to metaphor in the way it structures one concept in terms of another. This structuring is simply achieved in a different manner.

Metonymy is the use of "one entity to refer to another that is related to it." [43, p.35] An example of metonymy is THE CROWN FOR THE QUEEN. When people talk about the queen they often refer to her as "the crown." In this way an object related to the queen, her crown, serves as a reference to the whole queen.

The relationship between the two entities which take part in metonymy does not have to be of the part/whole kind. Indeed, Lakoff and Johnson note several different basic kinds of metonymy, which are then extended to be made more specific. THE PART FOR THE WHOLE involves identifying something with some subpart of it, as in the case of THE CROWN FOR THE

QUEEN, or THE FACE FOR THE PERSON. Another basic metonymy is THE PRODUCER FOR THE PRODUCT, allowing phrases such as "she owns a *Picasso*." In this case the relationship between the two entities is a *causal* one. Another causal relationship is evident in the base metonymy THE OBJECT USED FOR THE USER when someone says "the *buses* are on strike," for example. Other metonymies are CONTROLLER FOR CONTROLLED ("*Nixon* bombed Hanoi"), INSTITUTION FOR PEOPLE RESPONSIBLE ("*Caltex* raised its petrol prices again"), and THE PLACE FOR THE EVENT ("we don't want another *Vietnam*"). In each case, there is a strong relationship between the representing entity and the represented entity. This must be the case for a metonymy to function successfully.

Although metonymy is a different device to metaphor, Lakoff and Johnson suggest it is used in more or less the same way. This is because it is similarly systematic [43, p.37] and aids understanding like metaphor does [43, p.36].

Metonymy's systematicity is reflected in the discussion above, where it was observed that the same *base* metonymies are used repeatedly to structure the more specific metonymies that people use. Thus THE PART FOR THE WHOLE defines a whole system of metonymies for every day use such as THE FACE FOR THE PERSON or THE CROWN FOR THE QUEEN.

Metonymy aids understanding in an interesting way. In general, the entity chosen to represent in the metonymy is quite revealing about the entity represented, and what aspects are considered important. Thus, a metonymy such as THE FACE FOR THE PERSON, which is used when a picture of a face is shown as as representative of the person with that face, emphasises the importance we place on people's faces. A picture of someone's foot, for example, would *not* be considered as an adequate representation of a person. Similarly, the metonymy PICASSO FOR THE PAINTING, expressed in "she owns a Picasso," underlines that the importance here is that Picasso created the work and it is therefore prestigious, rather than the work itself necessarily being of chief importance.

3.8.2 The User-Interface

Metonymy is heavily utilised in the user-interface in the form of *icons*. Computer icons are typically used to represent actions the user can take in the system, or objects that metaphorically exist in the system. The former purpose, in particular, is often achieved through metonymy. The chief reason for this is that it is difficult to easily represent an action in a static environment such as a user-interface. Although animation is a possibility, it is far too distracting to be used to represent the many actions available to a user. interface. Metonymy represents actions well because of the various *causal* metonymies, as discussed above. In particular, computer icons often use the metonymy THE EFFECT FOR THE CAUSE, which is the basis of metonymies such as THE PRODUCT FOR THE PRODUCTION. Thus, the representation of the action "create a new document" in Microsoft Word is a small image of a blank document. This is the metonymy A BLANK DOCUMENT FOR THE ACTION OF CREATING A BLANK DOCUMENT or THE EFFECT FOR THE CAUSE. Metonymy thus gives designers a simple and efficient way of representing actions in a computer system.

An interesting feature of metonymies in computer systems is that the objects they refer to, such as the blank document in Microsoft Word, are not real objects, but objects in the metaphoric world of the user-interface. The blank document does not correspond to a *real* blank piece of paper, but to the user-interface metaphor THE DATA IS A DOCUMENT.

As discussed above, the choice of the representing entity in a metonymy reflects thinking about the represented entity. This is true in computer systems also, and is a key area of concern for designers using metonymic computer icons. As an example, consider the "paint" icon in a drawing program. The actual function here is not all that similar to real world painting, but the iconic image emphasises the painterly nature of the task. Thus

the emphasis is placed on the freedom of movement the tool provides, and its artistic nature, perhaps. Similarly, the magnifying glass, often used to represent a search function, emphasises certain qualities of searching. In particular, a magnifying glass emphasises the idea of close scrutiny, and poring over a document, which makes it more appropriate for searching in a word processor rather than, for example, an Internet search.

User-interface metonymy, as largely represented in computer icons, is a very important part of the design process. The choice of metonymies will influence how users think about the tools offered, albeit subtly. Thus it is advisable for designers to have a keen awareness of this device and its implications.

3.9 Conclusions

In this chapter the work of Lakoff and Johnson on the analysis of metaphor has been applied to the concept of the user-interface metaphor. In particular, three specific types of metaphor were identified: orientational, ontological and structural. Each of these metaphor-types were shown to be relevant when discussing metaphors in the context of user-interfaces. In each case, specific considerations were shown to be important, although some, such as cultural issues, were universal.

Relationships between the categories were also discussed in some depth. It was shown that the classes of metaphor lie on a scale of abstraction, with orientational metaphors being the most abstract and structural metaphors being the most specific. The concept of metaphorical entailments was then introduced to provide a means of indicating the structure or content of user-interface metaphors. The concept of metaphor novelty was expounded and shown to be an important consideration of user-interfaces, emphasising the necessity of user-testing when developing new metaphors and when utilising conventional ones. Finally, Lakoff and Johnson's discussion of the related device of metonymy was applied to the user-interface and shown to be particularly relevant to representing available actions.

This chapter has established a fairly detailed means of classifying and talking about the kinds of metaphors used in user-interfaces. The following chapter, which offers a structured and deep examination of the underlying structure of user-interface metaphors will compliment this view.

Chapter 4

A Semiotic View of User-Interface Metaphors

4.1 Introduction

This chapter presents a view of user-interface metaphors from the perspective of *semiotics*. This manner of examining user-interface metaphors will provide a more principled means of analysis for designers.

First, in section 4.2, a general introduction to semiotics is provided. Next, the basics of the theory of semiotics used in this thesis are outlined in section 4.3. A brief survey of the use of semiotics in computer science is then presented in section 4.4. A foreword about the difference between interpreting and generating signs appears in section 4.5, after which the main line of research in the chapter is commenced. In section 4.6, a plausible semiotic view of metaphor is presented. This is followed by an application of semiotics to user-interfaces in section 4.7. With these two semiotic approaches introduced, their combination is used to form a semiotic model of user-interface metaphors in section 4.8. The chapter is then reviewed and wrapped up in section 4.9.

4.2 Introduction to Semiotics

The most basic description of semiotics is that it is the study of signs. According to Charles Sanders Peirce, one of the founders of semiotics, a sign is "something which stands to somebody for something in some respect or capacity." [67, v.2 p.228] This definition makes it clear that a sign can be almost anything. Examples of the wide range of things that can be signs include footprints, smoke, written words, spoken words, computer images, and so forth; anything which can *mean* something to someone.

Semiotics is particularly concerned with the analysis of signs into parts, and how those parts interact to create meaning. This casting of semiotics as an analytical tool makes it useful for the detailed examination of signs. Naturally, semiotics can analyse *any* sign, but given that this thesis concerns signs *designed* for a user-interface, the notion of the *intentional* sign is the focus here.

It should be clear by now that it will be possible to provide semiotic analyses of user-interfaces. This is because user-interfaces are completely comprised of signs. Almost all of the elements in the user-interface are supposed to *mean* something to the user. The image of the trashcan on the screen stands for a method of file deletion and maintenance; the "okay" button in a dialog-box stands for the ability to accept what is suggested.

Semiotics comes in many forms, but the two main schools are the European school, largely

derived from the work of Ferdinand de Saussure [23], and the American school, which springs from the work of Charles Sanders Peirce [67] and Charles Morris [50].

The rest of this chapter will focus exclusively on the semiotics developed by Charles Peirce, and will apply it to user-interface metaphors. At the end of the chapter it will have been demonstrated that semiotics is a viable tool for understanding how signs in the user-interface function and, in particular, how user-interface metaphors can be modeled semiotically.

4.3 Basic Peircean Semiotics

The semiotics developed by Charles Peirce is a highly structured approach to the study of signs [67]. Because of this focus on structure, Peircean semiotics is an excellent tool for the analysis of signs in the user-interface. Computers are, by their nature, extremely structured machines, and the software run by them is equally structured. Peircean semiotics allows a particular kind of characterisation of that structure which can yield insight. Additionally, this brand of semiotics is extremely well-known and respected, and thus is acceptable as a base from which to extend.

By applying Peircean semiotics to user-interface metaphors, it is possible to harness a great amount of work done in the *general* field of semiotics. This means that for the investment of applying Peirce's view to user-interface metaphors, all other work on semiotics potentially becomes available for application also. This thesis will show how some of the analytical techniques available to semioticians prove useful in the analysis of user-interface metaphors. There is a huge body of work, however, and this research will only deal with a small, though fundamental, portion.

Before proceeding to discuss the semiotics of user-interface metaphors, it is important to gain a grounding in what aspects of Peircean semiotics will be used. The two key aspects of Peirce's work that will be focused on in this thesis are his division of the sign into a triad, and the concept of "unlimited semiosis." These will now be discussed in turn.

4.3.1 The Peircean Model of the Sign

Charles Peirce proposed a triadic model of the sign. In his view, the sign is divided into three parts: *object*, *representamen* and *interpretant*. These can helpfully be explained with an example. Figure 4.1 shows the Peircean view of the sign as applied to a common stop-sign.

As is shown in the diagram, the *representamen* is the actual embodiment of the sign. In this case, the sign in the world itself: a red octagon with the word "stop" written on it in white block letters. The representamen is the part of the sign that people encounter and attempt to understand the overall sign through. In fact, in much of Peirce's work, the terms "representamen" and "sign" are used somewhat interchangeably.

The *object* of the sign is the concept *cars must stop here*. This is the idea that the sign is meant to convey to those who encounter it. Other ways of expressing this are that the sign *means* that concept, or that it *represents* that concept.

Finally, the *interpretant* of a sign is the thought or concept that occurs in the interpreter's mind when they encounter it. Thus, in the figure, the person who has encountered the sign correctly thinks that they should stop their car. There is no guarantee, of course, that this "correct" interpretant will be arrived at. The person could have thought something like "I should stop smoking my cigarette now." Context and convention render this outcome unlikely, however.

As can be seen from the above explanation, the Peircean triad actually explains a *process* rather than a static set of parts of a sign. The process is known as *semiosis*, which Peirce described as "an action, an influence, which is, or involves, a cooperation of three subjects,

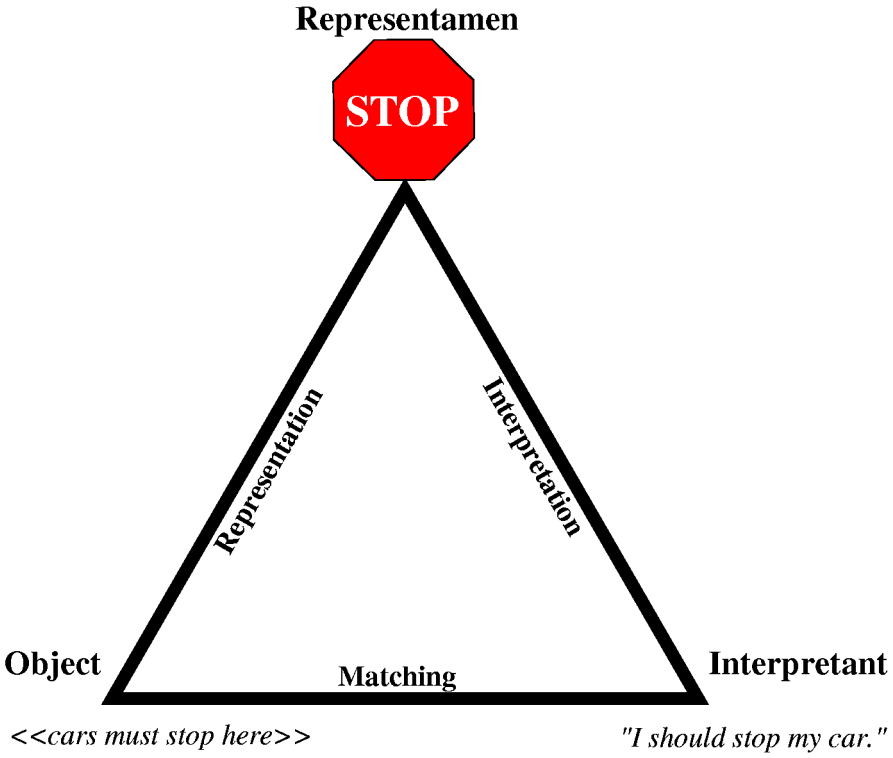


Figure 4.1: A diagram of the Peircean triad as applied to a stop-sign.

such as a sign, its object and its interpretant, this tri-relative influence not being in any way resolvable into actions between pairs." [67, v.5 p.488] This definition indicates Peirce's idea that the overall process is what is important and that it cannot really be decomposed further.

Despite this, it is instructive to consider what the relations between the parts of the sign might be. The three possible relations are shown in figure 4.1. These relations are unique to this thesis, and are *not* derived from the work of Peirce.

The *representation* relation occurs between the object and the representamen. It is clear that the representamen is meant to convey the object, and thus it should be considered as representation. This relation concerns how the object can be coded as a representation of some sort, be it visual, auditory, or otherwise. The representation relation for the stop-sign concerns the way in which the red-octagon and white word "stop" in some way means or represents the concept of stopping a car.

The *interpretation* relation is the link between the representamen and the interpretant. This can be thought of as concerning the process of an interpreter encountering the representamen and thinking about it, developing an interpretant. In the case of the stop sign, this relation would be a person seeing the stop-sign, and the thought-processes that lead them to think they ought to stop their car.

The final relation is that of *matching*, which occurs between the object and the interpretant. This can be thought of as a relation detailing how the interpretant, which is in the mind of the interpreter, relates to the object that the sign was representing. When the sign is an intentional one, this relation therefore concerns how *successful* the representamen was in conveying the object to the interpreter. In the case of the stop-sign, the matching relation was strong because the interpreter had the correct sort of thought in response to the sign. Had the interpretant been "I should never paint my car red" then it clear would not have matched the object well.

From the above discussion it can be seen that the Peircean triad provides a way of more

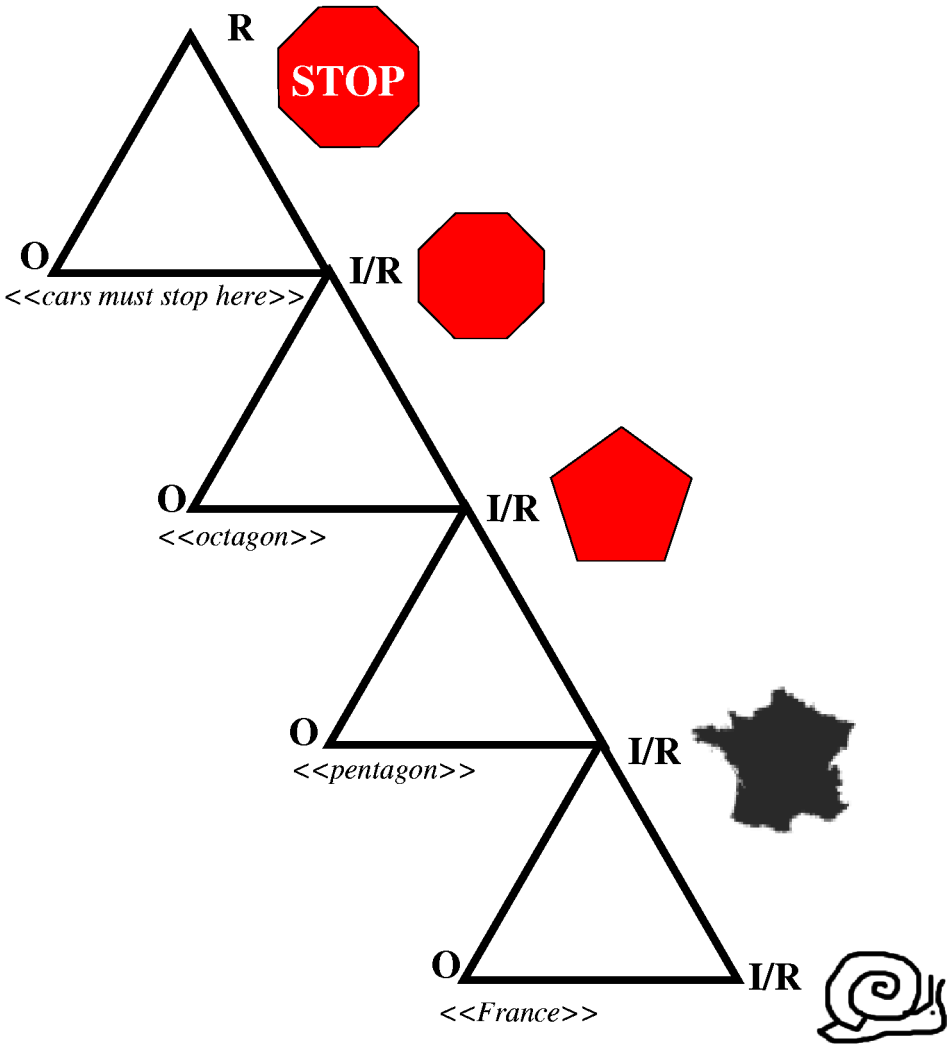


Figure 4.2: A diagram of the process of unlimited semiosis.

deeply analysing the signs that are constantly encountered in daily life. It becomes possible to say considerably more about them and provides a systematic vocabulary for discussion. This highly structural account of signs is what makes Peircean semiotics so appealing as a means for the analysis of computer-based signs.

4.3.2 Unlimited Semiosis

Before discussing the application of Peircean semiotics to the computer, it is important to examine the process of semiosis in a little more detail. In particular, it was suggested by Peirce, and developed further by Umberto Eco, that an encounter with a sign is not quite as neat as a single triad. In fact, in Peircean semiotics it is commonly thought that an encounter with a sign actually involves many interpretants, in a process known as *unlimited semiosis*. The term itself is due to Umberto Eco [29], but the idea is also apparent in Peirce's work. For example, Peirce describes a sign as "anything which determines something else (its interpretant) to refer to an object to which itself refers (its object) in the same way, this interpretant becoming in turn a sign, and so on ad infinitum." [67, v.2 p.300]

What this means in a basic sense is that the interpretant of a sign can be a sign itself. In particular, the interpretant of a sign can become the *representamen* of a new sign, with a

new object and interpretant of its own. It is clear from this description that the process can therefore go on forever. Hence Eco's terming it *unlimited* semiosis.

Figure 4.2 gives a possible example of unlimited semiosis. In this case, the initial sign is the sighting of a standard stop-sign. This makes the interpreter think of an octagon shape, which leads in turn to thinking about another polygon, the pentagon. The shape of a pentagon reminds the interpreter of the shape of France, and, when thinking about France, they think about the stereotypical eating of snails. Clearly, this could lead to further representamens and interpretants but the process should be apparent by now. Any interpretant can lead to further signs because the interpretant *itself* can represent something else. That is the basis of unlimited semiosis.

The chief interest in unlimited semiosis for the purposes of this thesis is the concept of *joining* two signs together. This use will be demonstrated after some discussion of the use of semiotics in computer science in general, and then the application of Peircean semiotics to both metaphor and basic user-interface signs separately.

4.4 Computer Semiotics

Although it has already been made fairly clear that semiotic analysis can be applied to computers as sign systems, there has not been as much work in the area as might be expected. There has still, however, been significant research devoted to the process of applying semiotics to computers. Because the most sign-intensive part of a computer system is the user-interface, a lot of work has focused on this area, although it is not exclusive to it.

Joseph Goguen has performed substantial research into the concept of an "algebraic semiotics." [34, 47]. The object of algebraic semiotics is to make semiotics a more rigorous and mathematical discipline. The chief application of algebraic semiotics has been to user-interface design, but it has additionally been used represent mathematical proofs. A key concept in Goguen's semiotics is the *semiotic morphism* which is intended to provide a representation in one sign system for signs from another system. Another emphasis of Goguen's theory is that signs ought to be studied within the context of *systems* rather than individually. The major point of Goguen's work on user-interfaces is an attempt to formalise their design by developing rigorous methods for mapping functionality to interface representations.

Peter Bøgh Andersen is likely the most prolific of computer semiotics researchers, and has produced large amounts of work focusing on the design of interfaces and computer systems in general. In his major work, *A Theory of Computer Semiotics*, Andersen develops an extensive theory on how semiotics can be applied to all aspects of computers [4]. The book includes an extensive case study of the application of his methods to a software system for a post office. Note that Andersen's semiotics is modeled from the Saussurean school, rather that from the Peircean semiotics used in this thesis.

Another important paper of Andersen's focuses specifically on the ability of semiotics to inform the programming of computers [3]. In the paper it is emphasised that computers are comprised of signs at every level and thus there are abundant opportunities to apply semiotic analysis. The focus is particularly on object-oriented programming.

Finally, in a short paper, Andersen also analyses whether semiotics is a good approach to human-computer interaction at all [5]. The major issues which Andersen focuses on are whether semiotics can help: make HCI more coherent; exploit insight from older media; define the characteristic properties of the computer as a medium; and place HCI into a broader context. In conclusion, Andersen seems to feel that semiotics is a useful tool, but that hopes should not be pinned too high on it solving all HCI problems.

An extremely concise treatment of Peircean semiotics can be found in the work of Mullet and Sano [52]. Their work is based on the more detailed approach of Mihai Nadin who

presents a view of computer system design through Peirce's triad [53]. Jean-Marc Orliaguet has provided an extremely detailed Peircean look at interfaces with the objective of "programming" the user's behaviour as far as possible [66]. Although his conclusion is that the issue has not really been solved, a significant, if complex, assessment of the applicability of Peirce to interfaces is provided. Clarisse de Souza analyses user-interface languages with the aim of aiding decision making during the design process [24]. In a paper by the present author a semiotic treatment of icons in the user-interface is provided, along with some proposed heuristics [9]. The focus in that paper is the analysis of icons based on Peirce's *categorisation* of signs, a matter not considered in this thesis.

As noted above, semiotics is also applied to matters aside from the user-interface. James Noble and Robert Biddle apply semiotics to the software engineering concept of design patterns [62], while Marcelo Pimenta and Richard Faust offer a semiotic approach to requirements gathering [68].

Overall, there has been some excellent research into the application of semiotics of various kinds to computer-systems, and especially to the user-interface. There is, however, clearly room for further work in the area. This is especially true of a semiotic approach to user-interface metaphors. The rest of the chapter is devoted to developing such an approach.

4.5 Interpretation and Generation

Before embarking on the development of a semiotic model of user-interface metaphors, it is important to discuss the notions of interpretation and generation. The key issue to raise here is that it is possible to look at the model of a sign from different stand-points. In particular, it can be viewed either as explaining an *interpretive process* or a *generative process*.

The interpretive process is the traditional understanding of the Peircean triad. Under this view, the model represents the process of a person understanding a sign. This places an emphasis on the interpretant of the sign. The generative process, on the other hand, concerns examining the triad from the point of view of the sign's designer. In this case, what is crucially important is the representamen and the object. In particular, the *designer* of a sign knows what the object of the sign is, and wishes to control the interpretant of the interpreter. This is the opposite of the interpretive view, in which the interpreter has their interpretant, and can perceive the representamen, but does *not* know the intended object.

Because this thesis focuses on user-interface design, the generative point of view can be considered the more important of these two. This is especially true because it is a good way of talking about *intentional* signs. The general process of creating an intentional sign is to have some object requiring representation. A representamen is then created to convey this object to an interpreter as an interpretant. The most useful fact about this process is that it introduces the concept of *correct* or *successful* interpretation. Because the designer of the sign intends to convey a *particular* object, the degree to which the interpretant matches that object is a measure of how *successful* the sign is as a whole. Clearly, when discussing user-interface metaphors it is very desirable to be able to say whether an interpretation of the sign is of the right or wrong sort, which is to say whether the interpretant successfully matches the object.

Despite the clear importance of discussing intentional signs, there is little literature on the subject. Umberto Eco provides some insight into the process, although he does not develop it into a rigorous model which could be presented here. In the remainder of this section some of Eco's ideas will be outlined in order to strengthen the case for viewing the Peircean triad from the standpoint of the sign *designer*.

The key concept we can take from Eco's work is that of the "Model Reader," originally developed in his book *The Role of the Reader* [28]. The Model Reader is the reader envisaged

by the author of a text who will understand precisely what it is that the author is trying to convey. Although Eco discusses this in terms of whole texts, the Model Reader concept can easily be applied to single signs also, especially since an entire text can be legitimately considered as a sign.

Eco writes that a text is "a device conceived in order to produce its Model Reader." [29, p.58] This means that, in addition to assuming a model reader, the author also creates a text or sign in such a way as to help in actually *producing* that reader.

The Model Reader concept is what helps to bridge the divide between the generative and interpretive approaches to a sign. Eco writes that "to make his text communicative, the author has to assume that the ensemble of codes he relies upon is the same as that shared by his possible reader. The author has thus to foresee a model of the possible reader (hereafter Model Reader) supposedly able to deal interpretatively with the expressions in the same way as the author deals generatively with them." [28, p.7]

Overall, then, this view of Eco's leads to a way of understanding how intentional signs might work. First of all, the author of the sign envisages some Model Reader, an interpreter of the sign who has certain beliefs and a certain context such that they will *correctly* interpret the sign. Note that this fits well with the desire to suggest that a sign might have a correct interpretation, which is claimed to be the sign's *object* in this thesis. Secondly, the author of the sign can seek to *produce* the Model Reader "through the use of given ... strategies." [29, p.128] Clearly, in the case of signs these strategies do not need to be solely linguistic. The point is that it is possible to help *create* a Model Reader through various forms of influence. It is also clear that this influence must take place through the representamen, which is the point through which author and reader are linked.

Although the above discussion does not provide any kind of rigorous *model* of the intentional sign, it does provide some justification for the view that will be held in this thesis. In particular, it is clearly justifiable to present a sign model that is from the point of view of a sign *author*. In this case, the interpretant of the sign can be seen as the interpretant of the Model Reader, rather than a particular reader. Because of this, the interpretant in such a model will be a representation of the *correct* interpretant of the sign, and will thus match the object of the sign. Additionally, Eco's work gives license to discuss *strategies* for producing the correct interpretant or Model Reader. Although the concept of strategies will not be utilised in this thesis, it is plainly a powerful idea to be developed in future.

The view of an intentional sign taken in this thesis, then, is as follows. The author desires to convey a particular *object* to an audience of readers or interpreters. In order to do this, a *representamen* is produced which is intended to produce a model *interpretant* which matches the object. In this way, a sign model can portray the intended effect of the sign, rather than one of the many *possible* effects. Note that, in order to actually produce the model interpretant, considerable thought as to who the audience of the sign is needed. This follows Eco's emphasis that the author must "foresee a model of the possible reader" [28, p.7]. This, in turn, ties in well with the traditional human-computer interaction principle of "know the user." [57, pp.73–78] By presenting the sign from the standpoint of the designer, it becomes possible to characterise their knowledge of the various parts of a designed sign. Although it is clearly not the whole story, this model does make the otherwise artistic and closed phenomenon of design more open to comprehension and discussion.

4.6 A Semiotic View of Metaphor

In order to provide a semiotic view of user-interface metaphors, it is first necessary to cast metaphor itself in the light of semiotics. By analysing metaphor in terms of Peircean semiotics it will become possible to combine it with a view of the user-interface sign. Surprisingly, there

is little formal work available on presenting metaphor using semiotics, although metaphor as a concept is frequently discussed in the semiotic literature.

One view, expressed by Thwaites et al. [79], is that metaphor is the interaction of *two* signs. Clearly, the two signs involved are the explaining concept, the *vehicle*, and the explained concept, the *tenor*. Despite a reasonable amount of discussion, Thwaites et al. never formalise the concept, and this makes it difficult to ascertain how or even whether the two signs are *linked* in some way. Additionally, the book follows the Saussurean school of semiotics and so is not immediately helpful in that sense either. Nonetheless, the notion that there are *two* signs involved is a useful insight.

In his book *Semiotics: The Basics*, Daniel Chandler expresses a view that seems largely compatible with that of Thwaites et al., and which is also Saussurean [18]. In his view, a metaphor consists of the signifier of one sign (like the representamen in a Peircean sign) combining with the signified of a different sign (like the interpretant of a Peircean sign). These two parts of the different signs are joined to become a *new* sign, which is the metaphor. This captures the concept of two signs being involved originally, but then suggests that they become a *new* sign of their own, with part of each sign being used.

What the above should make clear is that it is reasonable to consider a metaphor as a sign in itself. It may well be that it could also be decomposed into *separate* signs, but it is useful in this case to consider it as a single sign. This makes it apparent that the *representamen* of a metaphor concerns any way in which the metaphor is expressed, most obviously in language. Thus, the statement that "argument is war" is a possible representamen of the metaphor ARGUMENT IS WAR. Of course this is not the only possible representamen for that metaphor as other phrases could be used. Perhaps the more subtle phrase "he attacked my position" could also be considered a representamen of the metaphor. Additionally, it is perfectly possible for images and sounds to serve as the representamen in a metaphor sign.

Having established some concept of what the representamen of a metaphor might be, it is now important to consider what the sign's object is. The object is generally thought of as the meaning of the sign, or what the sign describes. There is substantial debate in the philosophical literature as to what a metaphor means and how it can be defined. The two chief approaches are the substitution view, often attributed to Aristotle [8] and the interactionist view pioneered by I. A. Richards [71] and extended by Max Black [10]. Under the substitution view a metaphor is effectively a brief way of predicating a number of properties belonging to one thing of another. Thus, a metaphor can be described as a set of properties of the vehicle that apply to the tenor. The interactionist view is more complex and suggests that a metaphor *creates* similarity, rather than simply bringing it to light [21, p.279]. Although both views are interesting, the substitution view turns out to the most useful in the case of semiotically describing metaphors, chiefly because it is the more simple and accessible.

Given that this thesis is strongly based in the work of Lakoff and Johnson, it is natural to look to their research for a means of describing the object of a metaphor. The answer is immediately apparent: *metaphorical entailments*. Lakoff and Johnson suggest that "metaphorical entailments characterize the *internal* systematicity of the metaphor ... that is, they make coherent all the examples that fall under that metaphor." [43, p.91] The point to be taken here is that metaphorical entailments are what characterise the metaphor. It is important also to note that the content or object of a metaphor is not simply a straight listing of metaphorical entailments, but also concerns "the way the entailments fit together." [43, p.150] Given this, the position taken in this thesis is that the object of a metaphor sign is a set of metaphorical entailments and the structure that binds them together.

The question remains as to *which* possible set of metaphorical entailments should be considered the object of a metaphor. The stand-point taken here is that this depends on the position from which the sign is being considered. Because the chief occupation of this

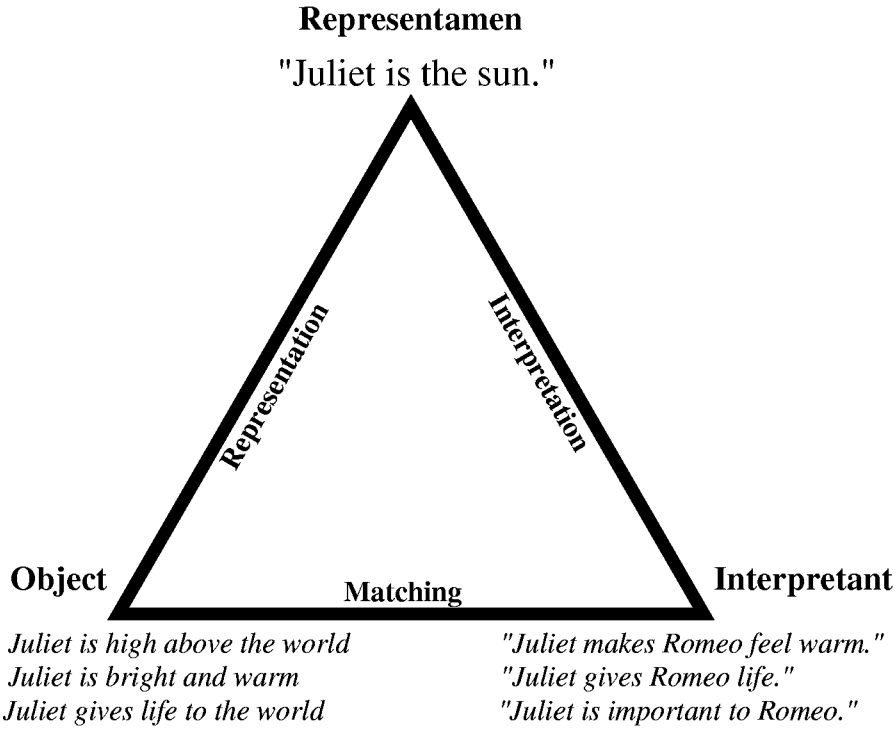


Figure 4.3: A semiotic model of metaphor.

thesis is to examine a designer's view of user-interface metaphor, the object of a metaphor is considered to be those metaphorical entailments taken to be the meaning of the metaphor by its author. This will be discussed in more detail when the full model of user-interface metaphor is expounded.

Before moving on, the concept of *semes* is interesting to note here. A seme can be considered as the smallest unit of meaning in some concept [35]. A seme of "bachelor" is "man," for example. An argument could be made that metaphorical entailments arise when there is a correspondence between the semes of the vehicle and the tenor. Thus, in the metaphor JULIET IS THE SUN, there can be claimed to be a match between the seme Romeo attributes to Juliet of making him feel alive, and the seme commonly attributed to the sun that it promotes life on earth. This leads to a metaphorical entailment "Juliet (like the sun) is a bringer of life." Although semes will not be explored further in this thesis, they are clearly an interesting means of further analysing metaphorical entailments.

Given that the object and representamen of a metaphor have been established, and that the interpretant of a metaphor is simply what the person encountering it thinks of, it is now possible to outline a semiotic model of metaphor. This model is presented in figure 4.3 using the standard example of the metaphor JULIET IS THE SUN.

The *object* of the metaphor sign is the collection of metaphorical entailments posited as the meaning by the metaphor's author. In the original case, this would be the set of entailments intended by Romeo, but each time the metaphor is used, the set of entailments that are its object change according to the person using it. In the example, the entailments are some of those thought of by the present author. This set of entailments is, in an important sense, what the metaphor actually *means* as presented in this context. In particular, it represents the things that the author of the metaphor wishes the interpreter to *think* when encountering the metaphor.

In order to create this impression, the *representamen* is used. As discussed above, the representamen is the means of conveying the metaphor to the interpreter. This can be in any

form, but with traditional metaphor it is normally through language. Hence, in the example, the representamen is "Juliet is the sun," a straight-forward statement of the metaphor.

Finally, the *interpretant* of the metaphor consists of the thought in the mind of the person who encounters the representamen. As indicated in the figure, this particular person has thought the right sorts of things about the metaphor, because they match fairly well with the metaphorical entailments. In this case, the interpretant belongs to the set of Model Readers.

The relations involved in a metaphor sign are quite straightforward. The object consists of the metaphorical entailments which define the meaning of the metaphor expressed as the representamen. Thus, the phrase "Juliet is the sun" is *representing* the metaphorical entailments of the object.

The *interpretation* relation is clearly the thought process undergone when an interpreter encounters the representamen. This involves them thinking about what it means for Juliet to be the sun, and possibly a conscious decision to interpret it metaphorically. The thought process yields an interpretant, which is the interpreter's response to the perceivable form of the metaphor.

The final relation is that of matching. In this case it concerns how well the concepts making up the interpretant match the metaphorical entailments of the metaphor which make up the object. Note that this only holds if the metaphor is used in order to produce a *particular* impression. In general, this is not true of more artistic metaphor, which instead produces a variety of impressions in its interpreters without the control of these impressions being overly important. For an intentional metaphor, however, it is possible to say how successful the interpretation is. This is based on how well the set of thoughts had by the interpreter match the set of metaphorical entailments intended by the sign.

Although, as already discussed, this is not the only possible model of a metaphor sign, it should be clear that it is a tenable one. In particular, it sensibly divides a metaphor into its constituent parts and relations and thus enables more structured discussion of the phenomenon. The model is intended to being useful in the context of human-computer interaction. Specifically, it is engineered to be linked with a model of a user-interfaces sign, which will be discussed in the following section.

4.7 A Semiotic View of the User-Interface

As with metaphor, the Peircean triad can be applied to the concept of a user-interface sign. Clearly, the images displayed in a user-interface are all signs: they represent or mean something to the user. Because of this, it is reasonable to expect that Peircean semiotics is relevant to the user-interface.

It is actually quite straightforward to create a triadic model of a user-interface sign. Much of this section utilises the work on a semiotic view of computer icons covered by the present author in a previous paper [9]. Figure 4.4 provides an example application of the model which will guide the following discussion. The example is of the print button presented by Microsoft Word which is clicked on in order to print the currently viewed document.

Obviously, the *representamen* of a user-interface sign is made up of the perceivable aspects of the interface relating to the underlying functionality represented. In the example, this consists of a button with a small, iconic image of a printer on it. Because user-interfaces are multi-media, these perceivable aspects can potentially involve any of the senses. While the most obvious senses utilised by a user-interface are the visual and auditory senses, there are others available also. In particular, the kinesthetic sense is engaged when physical movement is required in an interface, such as "dragging" a document into the trash. The additional senses of taste and smell, although unused, are also possible components of a user-interface

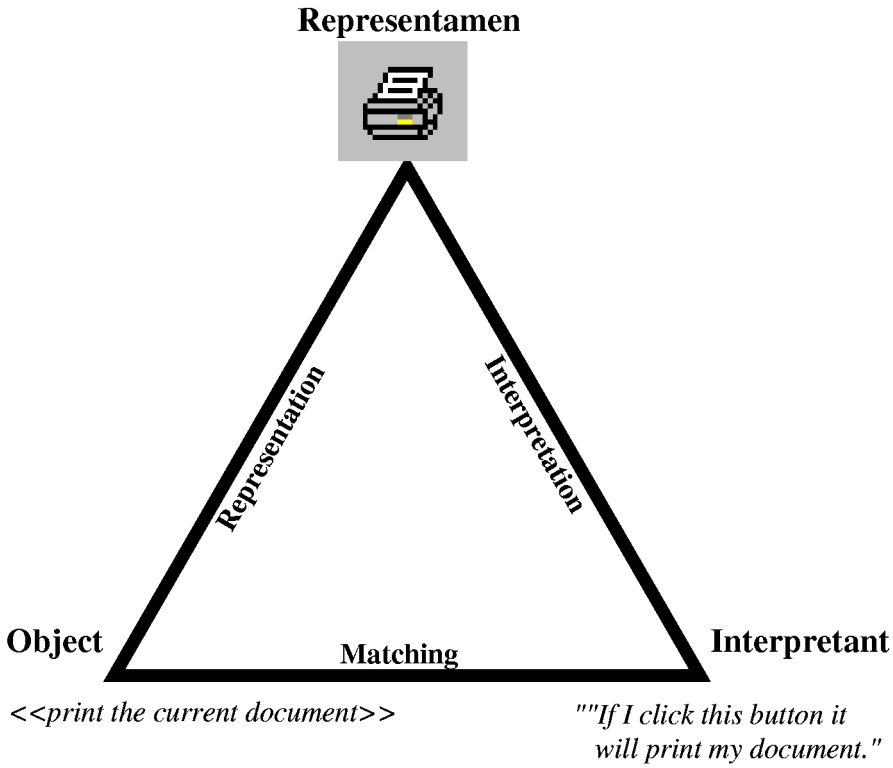


Figure 4.4: A semiotic model of a user-interface sign.

sign. As Mihai Nadin puts it, "we perceive signs through all our senses, and we generate signs that address the same. The fact that some of these signs (visual, auditory) are more important should not prevent us from considering any other sign that can be used for representation, communication, and communication functions." [53, p.269]

Note that, while it is possible to consider all the different aspects of a user-interface element as separate signs, it is more desirable to consider them all as part of the same sign. This makes semiotic modeling of interface elements considerably more simple and accessible as there is only one sign per object, rather than a multitude, one for each perceivable representamen.

The *object* of a user-interface sign is quite clearly the functionality that underlies that sign or representamen. In the example, the functionality is the ability to print a document. If the sign does not directly lead to a function, which is the case when the sign is not a button in the interface, the object can be considered as the properties represented by that sign. As an example consider the document icon frequently found in interfaces. This sign's object is the concept of a document on the computer: it means the concept *computer document* and the various things associated with that.

The *interpretant* of a user-interface sign is that which is thought by the user on encountering the sign. Because user-interface signs are *intentional* and because this thesis presents signs from the designer's point of view, the interpretant is that of the *Model User*. The Model User is the equivalent of Eco's Model Reader, but for user-interfaces. Therefore, the Model User is the user who "gets the right idea" about an interface, and reacts as desired. The interpretant in this case is something along the lines of "if I click on this button, it will print my document."

Once again, the relations discussed earlier can be applied to this user-interface sign. The *representation* relation concerns the process of deciding how to represent a particular piece

of system functionality with perceivable parts of the interface. In this case it concerns the designer's decision to have an iconic picture of a printer be representative of the print function. The link between representamen and object is quite clear here, and so the relation is fairly strong. Note that the link is provided by a *causal metonymy*, as discussed in section 3.8.

The *interpretation* relation concerns the desired process that leads to a correct interpretant. In this case it describes the process of seeing the icon of the printer and inferring from this that the button will print the current document. It can be seen as representing the ideal chain of reasoning which is undergone by the Model User. Because this relation represents a reasoning process it can help the designer examine exactly what inferences they are attempting to promote in the user.

The *matching* relation concerns how well the interpretant matches the object. When the interpretant is that of the Model User, it will match closely. Note, however, that the Model User could be substituted for a real user by performing user-testing and establishing the user's thoughts about the icon. In this case, the degree to which the interpretant and object match through this relation relation is a measure of success of the sign. Another example of generating an interpretant would be to have an expert who *simulates* a user. A method of doing this is called "studied ignorance," and is suggested as a part the usage-centred design testing of Constantine and Lockwood [19]. Additionally, user reviews can be used. This would aid in discovering user interpretations of the sign, and hence lead to possible analysis of the matching relation. User-centered design suggests a similar means of testing: observing users working with the software. While performing such observation, the testers could extrapolate plausible interpretants from the users' behaviour. Once again, these interpretants can then be examined via the matching relationship to establish interface success.

From the above discussion it should be fairly plain that the Peircean triad can be sensibly applied to signs in the user-interface. Although much further analysis could be performed at this point, the focus of this thesis is on user-interface metaphors, and so yet another step must be taken. In the following section the main point of this chapter will be put forward: a semiotic approach to user-interface metaphors.

4.8 A Semiotic View of User-Interface Metaphor

In order to create a semiotic view of a user-interface metaphor it is necessary to combine the previously discussed views of a metaphor sign and a user-interface sign. As noted in section 4.3.2, the concept of unlimited semiosis can be used to join together signs. Traditionally this transition from one sign to another occurs in the mind of an interpreter as they consider a sign, and find that their consideration leads to other signs. As this model is based on the designer's viewpoint, however, the joining of the two signs occurs through the designer instead. What is more, the designer is consciously aware of the joining of the two signs, as it is done *intentionally*.

The key point of unlimited semiosis to be utilised, then, is the notion that the interpretant of one sign can become the representamen of another. If this is considered as a conscious process, done intentionally, then it is similar to using one concept to fuel another in some useful way. This matches well with the idea of using a metaphor to fuel the design of some part of the interface. Therefore, the user-interface metaphor sign will involve a metaphor sign which is linked with a user-interface sign. This reflects the necessity of viewing a user-interface metaphor as a cohesive unit, while also recognising that it is divisible into a base metaphor along with a sign in the user-interface.

Figure 4.5 displays the model that involves linking the two types of signs already discussed. In particular, the interpretant of the metaphor sign becomes the representamen of the user-interface sign. This presentation of a user-interface metaphor as *two* signs linked together

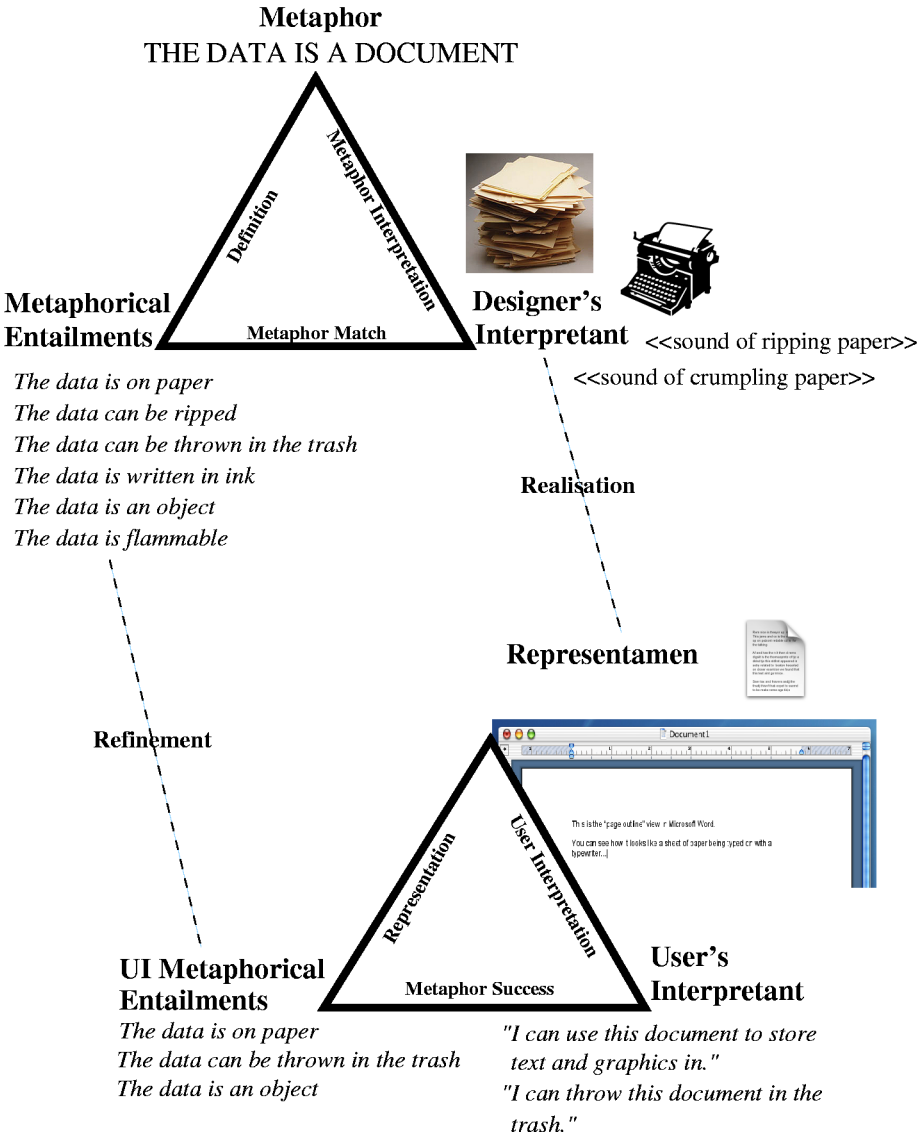


Figure 4.5: A semiotic model of user-interface metaphor.

allows discussion of both the surface level interface, and also the underlying metaphor. The various parts of the sign have been renamed to avoid confusion due to the fact there are now *two* signs involved. The following discussion of the model will be guided by the example of the *document* metaphor common to many modern user-interfaces. This example is reflected in figure 4.5, but note that the figure is not intended to convey a *complete* model of that sign.

4.8.1 The Parts of the User-Interface Metaphor

Metaphor

The representamen of the metaphor part of the sign is now simply called the *metaphor*, again because this is in keeping with Peirce's suggestion that the representamen *is* the sign in some sense. This part reflects the representation of the metaphor that the designer considers while creating the user-interface metaphor sign. It might be a sign in the designer's mind, or perhaps it might be written down in the project documentation. Note also that by having the representamen be the metaphor, this raises the possibility of having a lower level model

of the metaphor itself, possibly involving two separate signs, as discussed in section 4.6. This low-level representation could then be linked through its interpretant to the metaphor which is the representamen of the metaphor sign.

In the context of the document example, the metaphor is that THE DATA IS A DOCUMENT. The idea behind this metaphor is that, instead of viewing data input into the computer as amorphous, raw data, it can be cast as a document instead. The metaphor is especially relevant when the data entered into a computer serves a similar purpose as a written document. Although the data is clearly not *really* a document, the similarity of function allows the metaphor to function.

Metaphorical Entailments

The upper sign in the diagram is the metaphor sign, already discussed in section 4.6. The object of the sign is now referred to as the *metaphorical entailments* because those are literally what the metaphor is considered to mean. These are the entailments of the metaphor that the overall user-interface metaphor is based on. In particular, these are the entailments the *designer* believes the metaphor to have. They are also *independent* of the user-interface at this point, and are simply entailments of the metaphor in general.

In the context of the document metaphor, the metaphorical entailments are any considerations of documents that *might* be useful in metaphorically describing a collection of data. Thus a possible (and incomplete) list might be something like:

- The data is an object.
- The data can be written on.
- The data can be read.
- The data contains text, and possibly images and graphs, etc.
- The data can be ripped.
- The data can be typed up.
- The data is usually on white paper.
- The data can be photocopied.
- The data can be written in pencil or pen or ink.
- The data can (sometimes) be edited using twink or an eraser.
- The data contains information.
- The data can be set on fire.
- The data can be picked up and moved from place to place.
- The data can be thrown in a trashcan.

It is worth noting here that it is obvious not *all* of these entailments will be appropriate for use in the final user-interface. That is not what is important when discussing the metaphorical entailments, however, which can be seen more as a brainstorm about documents than anything else. Naturally, there are facts about documents which are not metaphorical entailments because they are simply inappropriate for application to the concept of data. This

includes entailments such as "the data can give you a paper-cut." An argument could be made that the metaphorical entailments are those entailments that the designer assumes will be shared by the user. This is similar to Eco's claim that "the author has to assume that the ensemble of codes he relies upon is the same as that shared by his possible reader." [28, p.7]

Designer's Interpretant

The final part of the metaphor half of the overall sign is the *designer's interpretant*. This reflects the result of the designer's consideration of the metaphor. It consists of the various thoughts the designer has about the metaphor overall, while thinking about how to design an aspect of the user-interface. These will be very much related to the metaphorical entailments which are the object of this sign. Note that this interpretant can include mental images, as well as concepts about motion, words, sounds and so forth. In fact, it can be considered as the overall mental reaction to the metaphor. What is more, the designer's interpretant could include any actual work done by the designer prior to actual implementation; it is the thought process which leads to the final implementation and encompasses all that design work.

In the example of the document metaphor, the designer's interpretant is any of the work done by the designer while attempting to figure out the representamen of the user-interface metaphor. Possible aspects of the designer's interpretant might be:

- The designer's mental images of documents, perhaps set down on paper or on the computer for documentation purposes. Specifically, this might involve images of typed documents, documents with images set in them, stacks of paper, and so forth.
- Thoughts and recordings of the sounds involved with documents, such as the sound of a typewriter working, paper tearing, or pieces of paper moving against each other.
- Actions involved with documents, such as typing one up, or throwing one into the wastepaper basket, for instance.
- Thoughts on the tactile sensations involved with documents, such as the feel of paper, or the feeling of hitting typewriter keys.

Note how all of these aspects of the designer's interpretant can be physically documented as a part of the design process. This makes the retroactive identification of the different parts of the user-interface metaphor sign considerably more simple. Note also that all of the aspects of the designer's interpretant can be linked back to the metaphorical entailments that were established. In fact, the designer's interpretant is, in some ways, realisation of those metaphorical entailments. By comparing the two an idea can be gained of the particular angle the designer took on the entailments of the metaphor.

Representamen

Because of the process of unlimited semiosis, the designer's interpretant now becomes the representamen of a new sign. This makes a considerable amount of sense in the context of interface design: the metaphor design is effectively the design process which leads up to the actual implementation of the user-interface metaphor. This implementation is the *representamen* of the user-interface sign discussed in section 4.7. The representamen is arrived at when the designer turns their interpretations of the metaphor into an actual user-interface element, and therefore can be considered as the realisation of that design. Here, the designer's interpretant, which is an interpretant of the metaphor sign, becomes the sign in the user-interface via implementation. The representamen consists of all perceivable aspects of a user-interface pertaining to the particular metaphor.

Figure 4.6: The standard document icon in MacOS X.

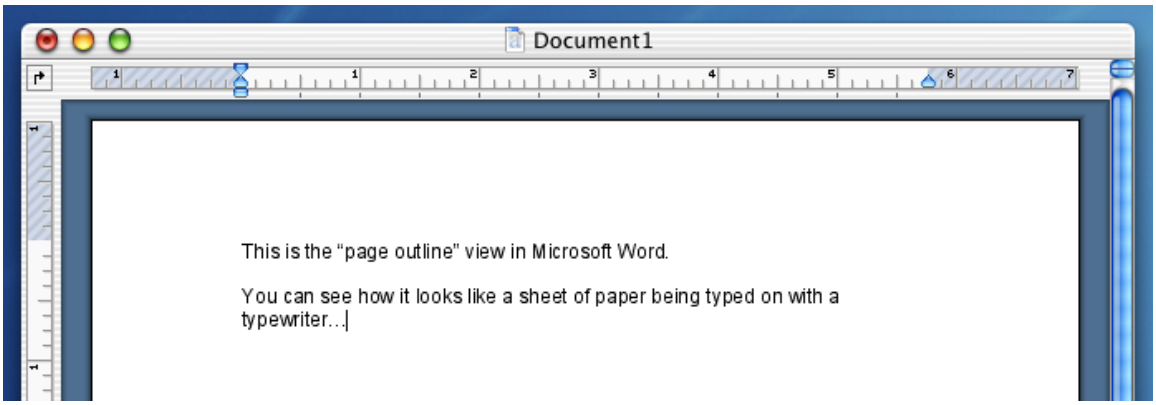


Figure 4.7: The standard view of the "paper" in Microsoft Word.

This can be made clear by considering the example of the document metaphor. As already discussed, the designer's interpretant for this metaphor involves the designer's ideas about the sorts of ways a document behaves, based on the metaphor. Thus, the representamen of the document metaphor consists of all aspects in the system that are used to represent these ideas. A few examples of the representamen are as follows:

- The document icon, as shown in figure 4.6. This visually represents the way that a document might look from afar, in a fairly abstract manner.
- The "Page Outline" view in Microsoft Word, which presents the current document being worked on as a sheet of A4 paper which can be typed on as in figure 4.7.
- The ability to pick up and move document icons on the desktop, reflecting the concept that documents are physical *objects*.
- The ability to "throw" a document into the trashcan.
- Explicit referral to collections of data as documents using language, as shown in figure 4.8.
- The ability to store a document in a folder.

Note how, in representing this metaphor, other possible metaphors are invoked, such as the trashcan and the folder. By combining metaphors, the overall illusion of metaphorical interaction is strengthened.

UI Metaphorical Entailments

Given that a new sign is now being discussed, there is therefore a new object to the sign. It is quite interesting to note that the upper (user-interface) sign mirrors the lower (metaphor) sign quite closely. In fact, the user-interface sign can be considered as the realisation of the metaphor sign in a different format: a user-interface implementation. To that end, the

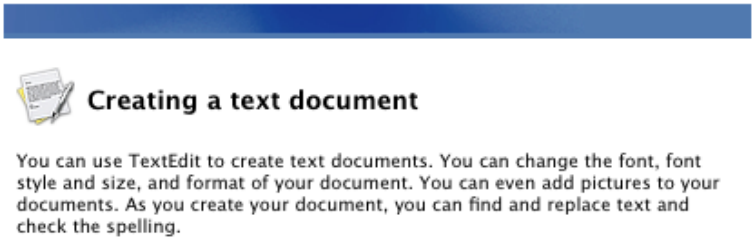


Figure 4.8: Example of explicit referral to a collection of data as a "document" in the MacOS X help system.

object of the user-interface sign is also a set of metaphorical entailments. This new set of metaphorical entailments will be called the *UI metaphorical entailments*, to distinguish it from the earlier set. The UI metaphorical entailments differ from the metaphorical entailments in that they are a set specifically tailored to the user-interface. In particular, they contain only those entailments that are relevant to the implementation of the interface. Basically, the point is that not all of the entailments of a metaphor will be appropriate for actual implementation. Therefore, the set of metaphorical entailments must be refined to the set of UI metaphorical entailments.

In the document example, not every one of the metaphorical entailments is useful for implementation. The designer realises this over the course of establishing the designer's interpretant component. Therefore, a new set of metaphorical entailments is established when the representamen is being created. These UI metaphorical entailments reflect the actually applicable entailments about documents which will work in the user-interface. A sample listing could be as follows:

- The data is an object.
- The data can be written on.
- The data can be read.
- The data contains text, and possibly images and graphs, etc.
- The data can be typed up.
- The data is usually on white paper.
- The data contains information.
- The data can be picked up and moved from place to place.
- The data can be thrown in a trashcan.

Note that various metaphorical entailments have been omitted in this listing. For example, entailments such as "the data can be ripped in two," and "the data can be set on fire" are no longer present because such details are not implemented. The complete set of UI metaphorical entailments comprehensively defines the functionality made available by the implemented document metaphor. Note, however, that they may not define *all* the functionality associated with the element in the user-interface. For example, it is possible to change the colour of the text in a document instantly and at will. This is *not* a metaphorical entailment because you cannot do such a thing to a real world document. The ability to combine metaphorical and non-metaphorical functionality is one of the powerful aspects of user-interface metaphors.

User's Interpretant

The final part of the user-interface metaphor model is the *user's interpretant*. As already discussed, this interpretant might be the Model User's interpretant, or a real user's interpretant, depending on the what the model is being applied to. If it is the Model User's interpretant, then the model represents the ideal process through which the sign goes, culminating in its successful interpretation. If a real user is used, then the model represents a kind of user testing, where the final interpretant can be compared with the UI metaphorical entailments to establish just how successful the sign was in conveying the correct way of interacting with it. Note also that the process of unlimited semiosis can be continued upward from the user's interpretant. In this case it can be used to model possible or actual thought processes undergone when interpreting the user-interface. In particular, it might be used as a means to show how the user must "reverse engineer" the perceivable representamen in order to establish the underpinnings of the sign, including the underlying metaphor.

For the document example, the Model User's interpretant will simply be a collection of thoughts which are similar to the UI metaphorical entailments. That is, the Model User will understand, through the representamen, the kinds of things they can expect to do with the user-interface via the document metaphor. This means they will think things such as "I can throw a document into the trash" or "I can type into this document." In a user-testing context, the user's interpretant will be established by the examination of real users. In this case, the interpretant must be somehow elicited either by observation, or by direct questioning, for example. In this case, the designers must have users interact with an implementation of the document metaphor, and then find out what it is they think. If the users think similarly to the Model User, then the metaphor is successful. If the users think things such as "I hope I don't accidentally knock this document off the desktop," then there are issues in the representamen to be resolved.

4.8.2 The Relations of the User-Interface Metaphor

The relations of this model of a user-interface metaphor are somewhat more complex than those already discussed in sections 4.6 and 4.7. This is because of the linking provided by the concept of unlimited semiosis. Not only must the interpretant of the metaphor sign be linked with the representamen of the user-interface sign in a kind of relation, but the objects of the two signs, both sets of metaphorical entailments, have a special relationship too. This section will outline each of the relations in turn and explain what they represent in terms of the design process.

Definition

Between the metaphorical entailments and the metaphor is the relation of *definition*. That is, the set of metaphorical entailments can be considered as the definition or meaning of the metaphor itself. This is reinforced by the position of Lakoff and Johnson, as discussed in section 4.6. In fact, they even go so far as to list a large number of metaphorical entailments for a metaphor and then write that these entailments "form a coherent whole as instances of the metaphor." [43, p.140] It can be taken from Lakoff and Johnson that the metaphorical entailments, along with their overall structure and coherence, define the content or meaning of a metaphor.

Thus, in the context of the example, the metaphorical entailments listed (along with the rest which complete the set) are the definition of the metaphor THE DATA IS A DOCUMENT.

Metaphor Interpretation

The relation of *metaphor interpretation* holds between the metaphor and the designer's interpretant. This relation represents the designer's thought process as they consider the metaphor and ponder what it means. The process leads to the designer's overall thoughts as embodied in their interpretant. This relation, therefore, concerns the designer's brainstorm as to what the metaphor might mean *in general* and independently of any strict design and implementation considerations.

In the example, this relation concerns the process of the designer thinking about the document metaphor, and then coming up with possible ideas for its use in the user-interface. This ultimately yields the designer's interpretant.

Metaphor Match

The *metaphor match* relation embodies how the designer's thoughts on the metaphor tie in to the metaphor's entailments. Because the designer effectively defines these entailments, the match ought to be quite close. It is important to note, however, that this relation can be actively examined. This can be done by have the designer's interpretant assessed for specific matches between it and the metaphorical entailments. This means that it can be discovered whether all of the entailments have been taken into account.

In the example, it can be seen that the ideas in the designer's interpretant do realise various of the metaphorical entailments. Thus, the aspect of the designer's interpretant which concerns the visual aspects of a document realises metaphorical entailments such as "the data is on paper," "the data can have a fold in it," "the data can have text typed on it," and so forth.

Realisation and Refinement

With the relations solely belonging to the metaphor sign now discussed, the extremely important issue of the relations *linking* the two signs involved in the model arises. There are two links here, one between designer's interpretant and representamen, and one between the two sets of metaphorical entailments. The *realisation* relation concerns the process of the designer's interpretant becoming a real implementation in a user-interface: the representamen. This is the process of actual implementation of the ideas about the metaphor. Thus, in the example, the realisation concerns the transformation of the designers concepts about how documents look and behave into actual interface elements and functionality.

The process of realisation is intricately linked with the parallel relation of *refinement*. The refinement relation concerns the process of narrowing down the set of metaphorical entailments to just those that will be true of the actual user-interface implementation: the UI metaphorical entailments. Because it is so tied to the implementation, this refinement will take place simultaneously with the realisation of the designer's interpretant as representamen in the user-interface. In the example, the refinement relation reflects the designer's consideration of the exact metaphorical entailments that will be used to define the functionality of the document metaphor in the user-interface. This is a culling process, where useful entailments, such as "the data contains text," are kept while inappropriate entailments such as "the data can give you a paper-cut" are jettisoned.

Representation

The *representation* relation concerns how the representamen links with the UI metaphorical entailments. Once again, this can be formalised by specifically indicating what these links are. In this way, it becomes possible to check on the coverage of the interface metaphor. That

is, it can be established whether all the UI metaphorical entailments are indicated in some specific way by the representamen, and whether the representamen indicates any non-existent entailments.

In the document example, the representation relation concerns how the implementation of the document metaphor conveys the UI metaphorical entailments already discussed. An example linking between the two is that visual representation of the document icon looks like a piece of paper. This ties to the UI metaphorical entailments which suggest a document can be moved, and even thrown in the trash.

User Interpretation

The *user interpretation* relation is the process of the user interpreting the representamen, as is traditional in Peircean semiotics. If the interpretant is that of the Model User then this interpretive process will be the ideal one, making all the correct inferences. If the interpretant is that of a real user, the user interpretation relation will have to be elicited from the user via interviews, thinking-aloud and other techniques.

In the document example, the user interpretation relation occurs while the user is interacting with the implementation of the document metaphor in the user-interface. The relation ends with the user's interpretation of how that metaphor functions.

Metaphor Success

The final relation is that of *metaphor success*. This relation concerns how well the user's interpretant is matched with the UI metaphorical entailments. In other words, this relation is a measure of how well understood the underlying meaning of the user-interface metaphor is by the user. The more the user has made correct inferences about how to interact with the system, the stronger this relation will be. Thus, it is a relation of how successfully the user interacts with the user-interface.

In the document metaphor example, this relation will concern how well the user has established the entailments about the DATA IS A DOCUMENT metaphor by interacting with the representamen. Instances where the user's interpretant matches with the UI metaphorical entailments indicate a degree of success of the metaphor. Instances where the user imagines functionality not present, or fails to see functionality which *is* present indicate a degree of failure. Generally, this relation indicates how well the representamen has conveyed the UI metaphorical entailments about documents to the user.

4.9 Conclusion

This chapter has outlined a detailed semiotic model of the user-interface metaphor. The model was reached by first explaining how semiotics works in general, in section 4.2, and then going into some detail on the model of the sign developed by Charles Peirce in section 4.3. Next, a brief literature review provided some insight into the present state of the application of semiotics to computers in section 4.4. An analysis of the differences and similarities between interpretation and generation in section 4.5 marked the beginning of attention to the user-interface specifically. The main task of the chapter was then begun, with discussion of a model of a metaphor sign (section 4.6) and a model of a user-interface sign (section 4.7) culminating in the model of a user-interface metaphor sign in section 4.8. The example of a document metaphor was used to illustrate the model in some detail.

The semiotic model represents the end of the purely theoretical investigation in this thesis. The next chapter recaps the material in this and the previous chapter as a structured

taxonomy. This is intended to provide a more easily accessible version of the concepts discussed so far. The remainder of the thesis then presents practical applications of the research such as a case study of the taxonomy (chapter 6), some usability heuristics (chapter 7) and an evaluation performed using those heuristics (chapter 8).

Chapter 5

A Taxonomy of User-Interface Metaphor

5.1 Introduction

This chapter presents the major points from the previous two chapters in a condensed, taxonomic form. In particular, it outlines the parts of the semiotic model from chapter 4 and also the categories of metaphor as discussed in chapter 3. A "taxonomy" is described in the Oxford English dictionary as being simply "the classification *of* something." [13]. The following division of a user-interface metaphor into its parts, and the broader classification of user-interface metaphors into types, therefore qualifies as a taxonomy. Each section will be presented in a structured format, which is as follows:

Title The name of the model part or category under discussion, i.e. "Structural Metaphor."

Brief Description A brief description of the concept in italic text.

Relevant Sections Sections in the thesis that deal with the topic in more depth.

Description A basic description of what the item under discussion is with relation to the user-interface.

Rationale Why it is useful to have such a category or part of the model.

Issues The major problems that could arise when dealing with the particular item.

Identification Thoughts on how to recognise instances of the item in an actual interface or design process.

Examples One or more examples of the item with some discussion.

Known Uses A brief listing of some of the common uses of the item in user-interfaces.

The format above owes some of its structure to the notion of Pattern Languages [2], and, more specifically, to Interaction Patterns [30, 31, 83]. Formal patterns are designed more with a specific problem resolution in mind, whereas the discussion here focuses more on a review of the material covered. Nonetheless, the basic format offered by patterns is a useful one for getting the information across in a structured way.

The basic divide of this chapter is clearly between the material on semiotics from chapter 4, which is presented first, and the material on Lakoff and Johnson from chapter 3, which is presented second. Before the semiotic material is covered, a basic diagram of the model will

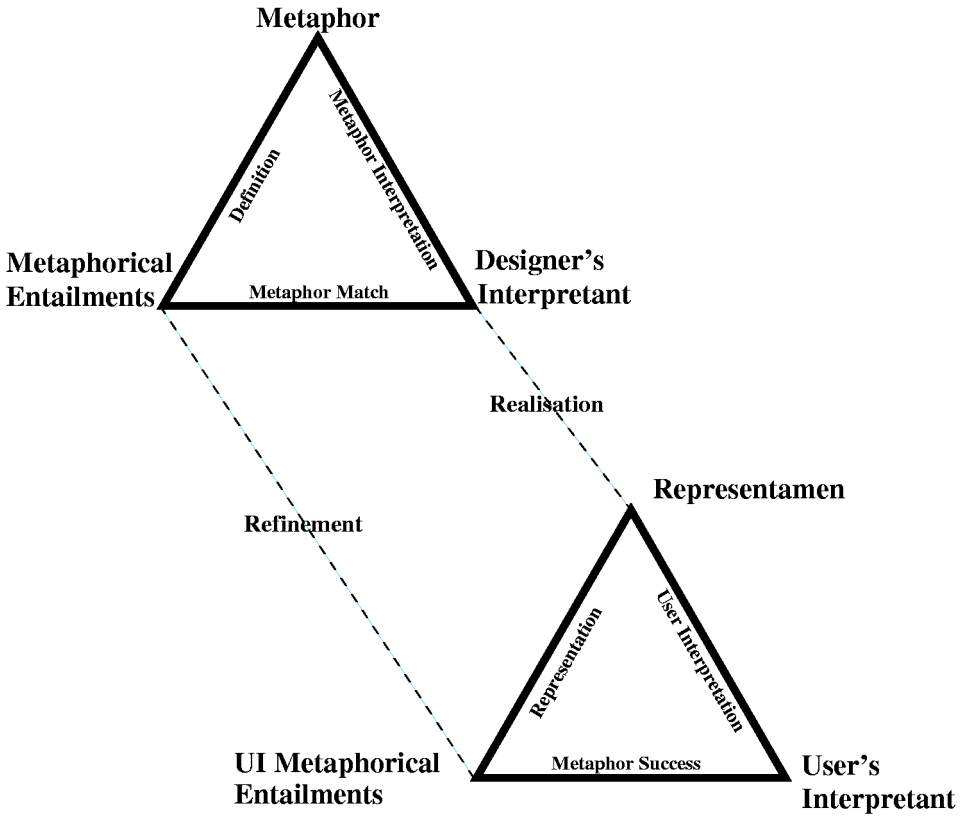


Figure 5.1: A semiotic model of a user-interface metaphor.

be produced to remind the reader of the overall structure involved. Similarly, in front of the material on Lakoff and Johnson, a diagram of the structure of the concepts will be shown.

5.2 The Taxonomy

5.2.1 Metaphorical Entailments

The metaphorical entailments of a user-interface metaphor are the set of entailments between the vehicle and the tenor that are suggested to explain the system concept.

Relevant Sections Metaphorical entailments are extensively discussed in sections 3.6 and 4.8.1.

Description The metaphorical entailments are those facts about the vehicle of the metaphor (the concept used to do the explaining) which are applied to the tenor (the concept to be explained). The set of metaphorical entailments forms a description of the metaphor and characterises its content or meaning. These metaphorical entailments are independent of actual implementation issues and can be seen as a listing of the facts about the vehicle that *could* be used. As stated multiple times in this thesis, the metaphorical entailments are not a complete transfer of vehicle facts to the tenor, but are instead a set of those that are deemed useful or explanatory.

Rationale Metaphorical entailments allow a designer to define in some detail what a metaphor means to them. The enumeration of the precise implications the designer at-

tributes to the metaphor are a valuable basis for discussion. The entailments can serve as a semi-formal statement of the designer's thoughts. Where more than one designer is involved, lists of metaphorical entailments can be combined to produce a more comprehensive set which the designers can then agree upon. This list can then also serve as a useful item of documentation of the metaphor design process.

Issues Because the meaning of a metaphor is quite personal, the set of metaphorical entailments varies from person to person. This means that two designers might disagree on the exact set of entailments. A sensible approach is to combine the metaphorical entailments envisaged by each member of the design team. This process of combination may well lead to useful discussion of the purpose of the metaphor to be used.

The second major issue involved with metaphorical entailments is that there can easily be a very great number. The number of identifiable facts about anything is vast, and possibly inexhaustible. Because of this, finding *all* the useful metaphorical entailments may not be possible. The best approach, then, is to have as many people as possible submit their list of metaphorical entailments to achieve the most coverage possible. It is also important to note that designers are seeking *useful* metaphorical entailments, rather than *any* metaphorical entailments. This narrows the search down considerably.

Identification Identifying metaphorical entailments involves careful consideration of the metaphor itself. In particular, designers must study the vehicle in detail and establish which features of it might be usefully transferred to the tenor. In this way, a list of entailments can be built.

Because the more general set of metaphorical entailments are not all included in the final user-interface, it is problematic or impossible to identify them retroactively when examining an interface. Instead, it would be necessary to examine the documentation created while the interface was being developed. If the designers did not follow the process suggested in this thesis, however, it will still be problematic to elicit the metaphorical entailments from this material.

Example The metaphor THE DATA IS A DOCUMENT has a multitude of possible entailments. Given that there are so many, a full list will not be presented here. Instead, a large range of possible metaphorical entailments for the user-interface metaphor will be listed, to give an idea of the sorts of things that can fill the definition above.

- The data is made of paper with ink on it.
- The data can be crumpled into a ball and thrown into the trash.
- The data consists of words, figures, and images.
- The data contains information.
- The data can be torn, or have coffee spilled on it.
- The data has a title.
- The data can have footnotes, a bibliography and so on.
- The data is often made of multiple pages of paper, perhaps fastened with a staple or paper-clip.
- The data can be photocopied to make new copies.

- The data can be mailed to other people in an envelope.
- The data can give you a paper-cut.
- The data has a particular length in words.
- The data can be confidential or public.
- The data is intended to be read.
- The data can be shredded.
- The data can be stored in a file-folder in a filing cabinet.

As can be seen, potentially any fact about real-world documents can be transferred to apply to the collection of data in need of explanation. By listing the metaphorical entailments is is possible to examine the basic ideas that might be applied in a written form. These can later be used to analyse the design and to discuss design choices with others.

5.2.2 Metaphor

The metaphor component of a user-interface metaphor is the actual statement of identity used to explain the system concept.

Relevant Sections The concept of a metaphor has been discussed extensively in this thesis, including a general introduction in chapter 2, the philosophical analysis of Lakoff and Johnson in chapter 3, and semiotic analysis in chapter 4.

Description The metaphor is quite simply the concept on which the design of a user-interface metaphor hinges on. It is the identification of a system concept to be explained with another concept which will help to explain it. A metaphor is most easily represented by a textual statement such as X IS Y, where X is a system concept to be explained, and Y is an explaining concept. The metaphor is described by the metaphorical entailments discussed above.

Rationale Clearly stating the metaphor to be used in the user-interface is a crucial part of designing a user-interface metaphor. Without this concrete statement the design process could proceed without an exact idea of what is being developed. The metaphor allows a focus on the particular concepts involved and provides documentation.

Issues Choosing the correct statement of a metaphor can be a challenging task. There are, perhaps, subtle differences between the metaphors THE DATA IS A DOCUMENT and THE COLLECTION OF BITS IS A DOCUMENT, for example. Additionally, there are *certainly* major differences between the metaphors THE DATA IS A DOCUMENT and THE DATA IS A VIDEOTAPE. The key to defining the metaphor is making sure all those involved in its design are clear on what is meant, so that they can work in harmony. The exact choice of the vehicle is absolutely crucial, but the choice of the tenor should not be discounted either.

Identification Retroactively identifying an existing metaphor is achieved by examining the interface for its metaphorical entailments, and reasoning from these entailments to what the metaphor must have been. To establish the tenor of the metaphor, it is best to consider what aspect of the underlying system the entailments seem to be explaining. Thus, if all of the apparent entailments deal with the removal of files, the tenor of the metaphor is likely concerned with file deletion. To establish the vehicle of the metaphor, it is necessary to examine the collection of entailments and to reason about what sorts of concepts involve that particular set of entailments. Often the vehicle may be established by examining the representation of the metaphor in the system (the representamen), and directly taking note of the vehicle used.

Example In the case of the continuing example the metaphor is simply the statement that THE DATA IS A DOCUMENT. This, coupled with the metaphorical entailments above, provides a metaphor and its definition. This can serve as the basis for the design process for a user-interface metaphor.

5.2.3 Designer's Interpretant

The designer's interpretant of a user-interface metaphor is the collection of thoughts the interface designer has about the chosen metaphor.

Relevant Sections The designer's interpretant is discussed explicitly in section 4.8.1, while interpretation is more generally discussed in section 4.5.

Description The designer's interpretant is a means of referring to the designer's thought process about the metaphor when considering how to design its representation in the user-interface. While in semiotic terms an interpretant tends to be strictly in the mind, the designer's interpretant might also be extended to cover any physical manifestations of the thought process. This could include drawings, written brainstorms, or post-it notes, for example. This interpretant can include mental or actual images of how the metaphor might look when implemented, as well as considerations of other representation techniques such as sound, interaction and animation. The designer's interpretant can be linked to the metaphorical entailments to establish which of them the designer considered to be useful on further consideration. It is, first and foremost, the designer's reaction to the metaphor. That said, it is a *directed* reaction, and part of the design process.

Rationale Formally acknowledging the designer's thought process about the metaphor can help to bring the creative process firmly into the overall design process. By establishing this aspect of the user-interface metaphor model it should become easier to discuss important parts of the creative process, and also to label particular design artifacts as belonging to this stage. The designer's interpretant also emphasises creative thinking by designers *independent* of actual interface implementation. Maintaining records of the artifacts related to the designer's interpretant can aid analysis of the thought process and justification of decisions if aspects of the design need to be revisited later on.

Issues Given that interpretants are most often in the mind, the designer's interpretant could be difficult to capture in any real sense. As noted above, however, certain artifacts of the design process can be seen as representing the designer's interpretant, and so it is not completely impossible to document this aspect of the model.

Identification Identifying the designer's interpretant amounts to considering all creative work done *after* a metaphor has been chosen, but *before* any implementation specific work has been performed. As soon as implementation details arise, the interpretant is no longer in the forefront of the process. A particular manifestation of the designer's interpretant is any research performed by the designer on the vehicle of the metaphor. Collections of pictures and analysis of the vehicle are important representations of the interpretant.

Example There are clearly many ideas to be had about the metaphor THE DATA IS A DOCUMENT. Some of these ideas will be images, some sounds, and some interactive; some will be abstract and some concrete. A possible set of ideas will be listed here, to give some indication of the kinds of things that can be included as the designer's interpretant.

- Drawings of what different sorts of documents look like: Official documents, personal documents, etc.
- Lists of the sounds associated with documents: when you type up a document on a typewriter, when you screw up a piece of paper to throw it away, when a piece of paper is shifted on a desk, etc.
- Thoughts about the different weights of documents: a large bound volume as compared to a single sheet of paper.
- A video of people using documents in typical ways: typing them up, mailing them, putting them in in-boxes, throwing them away, etc.
- Lists of the sorts of things put in documents: Such as images, text, graphs and so on.
- Lists of the different type-styles used in documents: bold, italic, underlined, etc.
- Lists of ways of storing documents: folders, envelopes, in-boxes, out-boxes, etc.

This list gives some idea of the vast array of things that can form a part of the designer's interpretant. As can be seen, it is effectively a brain-storm about the vehicle at this point, but with an eye towards application to the tenor.

5.2.4 Representamen

The representamen of a user-interface metaphor is the collection of perceivable aspects of the user-interface which communicate the metaphor to the user.

Relevant Sections The concept of a representamen of a user-interface metaphor is addressed in section 4.8.1.

Description The representamen is the most easily identified of the parts of a user-interface metaphor because it is the actual result of the design process and the aspect of the metaphor which is perceived and interacted with. The representamen can involve any of the various senses, but most typically involves the visual, auditory and kinaesthetic. It results directly from the designer's interpretant discussed above and can be seen as the realisation of the interpretant in the user-interface. The representamen is tied strongly to the UI metaphorical entailments which it is intended to communicate to the user.

Figure 5.2: The traditional document icon from MacOS X.

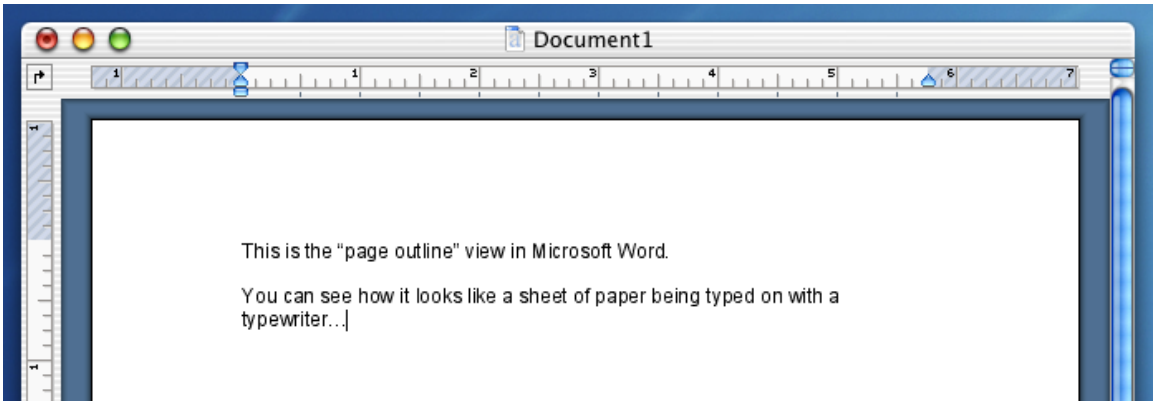


Figure 5.3: The standard "paper" interface to Microsoft Word for MacOS X.

Rationale In any model of a user-interface metaphor (or any aspect of a user-interface) there must be some means of referring to the actually perceivable aspects. The representamen serves as the link between the designers of the interface and its users, and is, in this way, the key to the entire model.

Issues Identifying all parts of a representamen could be difficult because it can involve any of the senses. In particular, noticing metaphorical representation with senses other than the visual and, perhaps, auditory is an uncommon task. Great care, therefore, should be taken when examining an interface to identify the representamen of a metaphor. The best approach, it seems, would be to consider each of the possible means of representation in turn, and to check whether the interface presents the metaphor using that technique. Additionally, knowledge about the actual metaphor and its metaphorical entailments would greatly assist in the recognition of their representation in the representamen.

Identification As noted above, identification of the representamen is likely the largest issue involved with it. By considering in turn the various senses, however, it should be possible to detect the various aspects which make it up. Aspects to consider involve graphics, animation, sounds, interaction, language and so on. Any part of the interface is potentially an aspect of the representamen for a user-interface metaphor.

Example There are various examples from the user-interface that help make up the representamen for the metaphor THE DATA IS A DOCUMENT of which figures 5.2 and 5.3 are two. Figure 5.2 shows the traditional document icon from MacOS X. This representamen indicates the document icon quite directly by having an iconic image of a document represent the collection of the data on the hard-drive. Features such as the folded down corner to indicate the document is made of paper, and the indication of writing, to indicate it is paper with content, show how aspects of the designer's interpretant have been used in the representamen. Figure 5.3 shows the interface to Microsoft Word in the "Page Layout" view. In this case it can be seen that the user types onto what appears to be a sheet of blank paper.

Additionally, using a keyboard to input one letter at a time recalls the use of a typewriter. These features once again are the realisation of aspects of the designer's interpretant, and are intended to communicate particular metaphorical entailments, as discussed next.

5.2.5 UI Metaphorical Entailments

The UI metaphorical entailments of a user-interface metaphor are those entailments of the metaphor which are relevant to the functionality represented by the user-interface.

Relevant Sections UI Metaphorical Entailments are discussed explicitly in section 4.8.1, while metaphorical entailments are more generally addressed in section 3.6.

Description The UI metaphorical entailments form the set of entailments of the metaphor which are applied in the implementation of the user-interface. They describe the actual functionality that the metaphor provides access to and explanation of. Thus, the UI metaphorical entailments form a semi-formal description of the functionality afforded by the metaphor and, as such, are an important part of the documentation of the user-interface. It is important to note that the UI metaphorical entailments do not describe *all* the functionality in the user-interface, but only that functionality addressed by the user-interface metaphor. They are represented through the representamen and can be considered as the *meaning* or *intention* of it.

Rationale The UI metaphorical entailments form a valuable means of characterising exactly what the metaphor is intended to convey in the context of the user-interface. That is, the UI metaphorical entailments form a description of what it is the user is intended to think upon their encounter with the representamen.

Additionally, these entailments provide a way of analysing the representamen in detail by checking that each UI metaphorical entailments is represented and that *only* the UI metaphorical entailments are represented. Similarly, the entailments provide a way of analysing whether test users have gained the appropriate concepts about the interface, because they describe the intended meaning.

Issues As with metaphorical entailments, it can be difficult to establish a definitive set of UI metaphorical entailments. That said, it should be considerably more straightforward, given that they are a subset of the metaphorical entailments. Establishing this set of entailments well is very important because it forms a semi-formal specification of the user-interface metaphor.

Identification Identifying the UI metaphorical entailments can be a difficult task. In fact, it is the task asked of a user when they are presented with the interface. The best way to identify the entailments is by examining the interface's various means of representation. Thus, the representamen of the user-interface metaphor must be studied because it is the key link to the UI metaphorical entailments. This means considering all the senses catered to in the user-interface.

Additionally, paying close attention to the functionality afforded by the representamen is important. This will help because a key aspect of the metaphorical entailments is that they define the interactive possibilities of the metaphor.

Example For the example of THE DATA IS A DOCUMENT only certain of the metaphorical entailments discussed earlier have been transferred to the user interface. These entailments are descriptive of the functionality offered by the document metaphor. Once again, however, the number of entailments is great, and so not all of them will be listed here.

- The data is made of paper with ink on it.
- The data can be crumpled into a ball and thrown into the trash.
- The data consists of words, figures, and images.
- The data contains information.
- The data has a title.
- The data can have footnotes, a bibliography and so on.
- The data can be mailed to other people in an envelope.
- The data has a particular length in words.
- The data can be confidential or public.
- The data is intended to be read.
- The data can be stored in a file-folder in a filing cabinet.

5.2.6 User's Interpretant

The user's interpretant of a user-interface metaphor is the mental result of the user's encounter with the representamen.

Relevant Sections The user's interpretant is discussed explicitly in section 4.8.1, while interpretation is more generally discussed in section 4.5.

Description The user's interpretant arises when a user has encountered the representamen in the user-interface. The result of this encounter will be various thought processes, including considerations of how to interact with the representamen. Interaction will, in turn, lead to more interpretants, based on further insight gained into the interface.

The user involved can be of at least three types. Firstly, the user might be the Model User discussed in section 4.7. In this case, the user's interpretant will be the ideal one, where all of the UI metaphorical entailments are grasped and an ideal reasoning process is performed. Secondly, the user might be a theoretical user, invented by the designers to test the interface. In this case the interpretant might be made up during a cognitive walk-through of the interface [84]. Finally, the user might be an actual user involved in user-testing. In this case the user's interpretant will need to be established by examining the user's behaviour and also directly via questionnaires or interviews.

Rationale In any model involving the user-interface, the user is a key component. The user's interpretant provides an excellent means for including the user when considering user-interface metaphors. The ability to fit different views of the user into the model, as noted above, is a significant advantage.

When the user's interpretant is established it can be compared with the UI metaphorical entailments to provide an analysis of how well the user's interpretation matches with the

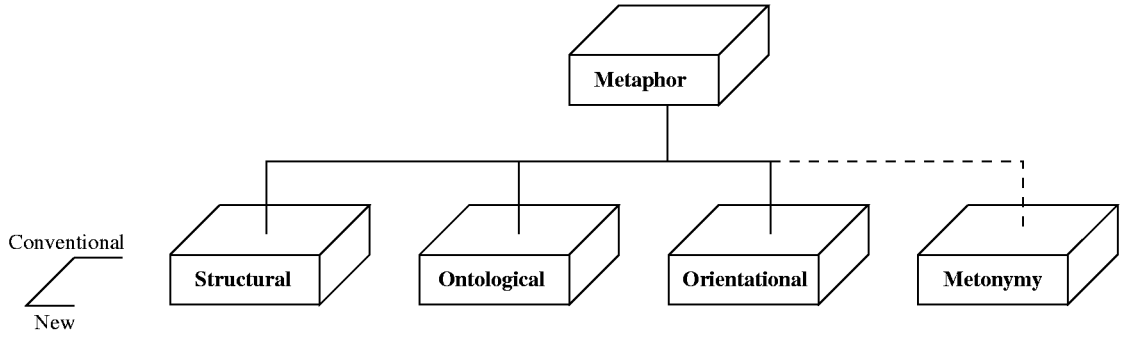


Figure 5.4: A diagram of the general structure of the Lakoff and Johnson material.

designer's vision. In this way, particular shortcomings and achievements of the metaphor can be specifically analysed with reference to the UI metaphorical entailments and the representamen.

Issues The particular difficulties here concern identification once again. In particular, establishing the interpretant of a real test-user could be very difficult. Finding the exact interpretant would require mind-reading, and so other, more traditional, techniques for establishing a person's opinions and thought-processes must be used. Simple examinations such as questionnaires and interviews, as well as the thinking-aloud technique should yield a fairly good view of the user's interpretant.

Identification As discussed already in this section there are several types of user that can be considered with reference to the user's interpretant. The interpretant of the Model User can be established by the designers or testers attempting to come up with the ideal thought processes for interacting with the user-interface metaphor. The theoretical user's interpretant can be found by the testers simulating a particular kind of user as they go through the interface themselves. Finally, the interpretant of a test user can be established using the techniques already discussed.

Example The user's interpretant of the metaphor THE DATA IS A DOCUMENT depends, as already discussed, on the kind of user under consideration. Having interviewed a test user, it might be discovered that they think they ought to be able to shred important documents they want to definitely remove from their computer, for example. The designers, on performing a cognitive walk-through, might find that they feel a real user would be confused by the ability to change the colour of a document's icon, and remove that functionality. The Model User, naturally, will have all the right ideas about the user-interface. Thus, the Model User will realise that documents can be stored in folders, and thrown in the trash, but that it they cannot give paper-cuts or catch on fire.

5.2.7 Orientational

An orientational user-interface metaphor is one which identifies a system concept with a spatial concept such as "up" or "in front."

Relevant Sections Orientational metaphors are discussed extensively in section 3.2.

Figure 5.5: The standard navigation controls in a "wizard" dialog box.

Description In the user-interface, orientational metaphors identify system concepts with spatialisation concepts. This is intended to allow users to understand an aspect of the system via their natural understanding of the physical world, specifically its spatial nature. In every day life, spatial concepts are used when understanding navigation, quantification, importance, and much else. In the user-interface, orientational metaphors are used for similar purposes, but especially the three just mentioned.

Rationale Orientational metaphors are considered useful because they leverage the very basic concept of space. All humans are extremely familiar with the notion of physical space, because it is a fundamental part of existence in the real world.

Issues Orientational metaphors can be significantly cultural in nature. This may prevent them being understood between cultures which is an issue for international software. In particular, different cultures may use different orientational metaphors. In this case, confusion may arise as to what a particular orientation means in the interface.

Because of their fundamental nature, orientational metaphors may be overlooked in the user-interface. The concept of spatialisation is so natural to humans that it is not necessarily recognised as metaphorical when applied to software design. Designers must be wary of unconsciously utilising orientational metaphors chiefly because of the cultural issues discussed above.

Identification Orientational metaphors can clearly be identified by looking for spatialisation concepts in the user-interface. An example is the ability to move interface elements, such as sliders, in a direction to have some effect on the system. The typical spatial concepts used are:

- Left and right.
- Front and back.
- Up and down.
- Over and under.

On finding instances of these concepts in an interface, identifying orientational metaphors should be possible by considering what it is that the concept means or indicates in the context of the system.

Examples Figure 5.5 indicates a navigational use of orientational metaphor. The "back" and "forward" buttons embody the metaphors REGRESSION IS TO THE LEFT and PROGRESSION IS TO THE RIGHT respectively. In this case the concept of left is identified with moving *backwards* through a series of dialog boxes, and vice versa for the concept of right. This orientational metaphor is likely based on the activity of reading from left to right, a highly *cultural* basis, or on the notion of a time-line on which time moves from left to right.

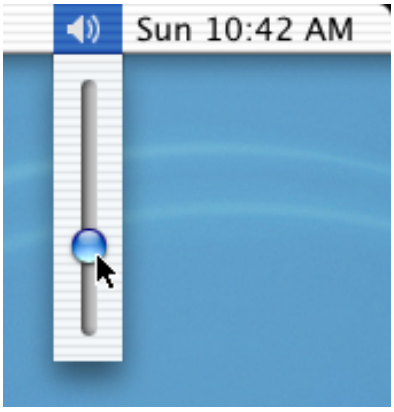


Figure 5.6: The vertical slider used to control volume in MacOS X.

Figure 5.6 demonstrates a quantificational use of orientational metaphor. In this case the ability to move the slider up and down to change the *amount* of volume uses the metaphors MORE VOLUME IS UP and LESS VOLUME IS DOWN. This likely springs from the physical experience that adding more to a pile or container increases the vertical level of material and is simply a more detailed version of the MORE IS UP metaphor.

The traditional practice of having active windows overlap inactive windows in a user-interface is an example of the metaphor ACTIVE IS NEAR or perhaps IMPORTANCE IS NEAR. The overlapping is intended to give the impression that the active window is *closer* to the user than the other windows. Traditionally in the real world we focus on those things closest to us as being important. When we wish to examine something we move it closer to us, or move ourselves closer to it.

Known Uses Scroll-bars. Dialog boxes. Wizards. Browser controls. Sliders.

5.2.8 Ontological

An ontological user-interface metaphor is one in which a system concept is identified with a basic concept of every day life such as substance, object or entity.

Relevant Sections Ontological metaphors are addressed in depth in section 3.3.

Description Ontological metaphors identify system concepts with the basic categories of existence in the physical world. This means concepts such as "object," "substance," "entity," "container," and so forth. The explaining concepts are not actual existent things, but are instead the concepts we use to *categorise* the things in the world. Ontological metaphors in the user-interface are typically used for referring, quantification, and indicating causation, as discussed in section 3.3.

Rationale Ontological metaphors are intended to help users understand system concepts through their fundamental understanding of the natural classes of things in the world. Particularly, they are intended to allow the users to use their basic knowledge of how the world physically functions to negotiate the user-interface. Humans have extensive experience in how the basic categories of existence work, and this knowledge is hoped to transfer to an understanding of system concepts identified with those categories.

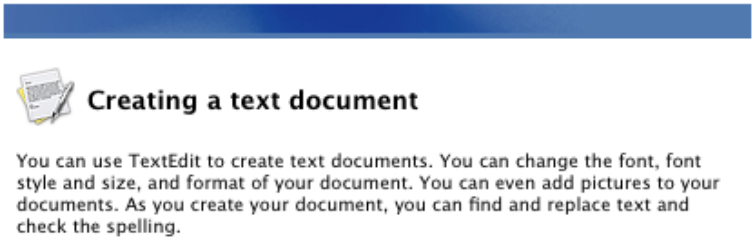


Figure 5.7: An example of referral in the user-interface.

Issues Much like orientational metaphors above, ontological metaphors are often used unknowingly because they are so basic. For example, a text-box, which seems very far removed from anything concerning the real world, exhibits the qualities of a *container*, which makes it an ontological metaphor. Additionally, ontological metaphors are present beneath almost every *structural* metaphor, and so can be used unknowingly in this way too. When this takes place, it is possible that there are implications from the ontological metaphor about the underlying system concept that are undesirable within the function of the system. Designers must therefore be aware of their usage of ontological metaphors, to be sure that they convey the correct messages to the user.

A case might also be made that ontological metaphors are used too enthusiastically. By focusing on making a user-interface behave similarly to the physical world through ontology, it is possible that *better* design solutions are missed, and that system concepts are forced to conform to awkward standards.

Ontological metaphors can also be quite difficult to identify because they are not necessarily easily perceivable within the system. Ontological concepts such as object and entity on their own do not readily lend themselves to images or sounds, for example.

Identification Ontological metaphors are likely the most difficult of the categories of metaphor to identify, as mentioned above. In general, the best way to identify them is through experience and practice. Some of the typical ontological concepts used in interfaces are as follows:

- Objects.
- Entities.
- Substances.
- Containers.

Additionally, looking for aspects of the system that seek to represent the purposes of ontological metaphor discussed above should aid in their identification. When a process of causation is apparent, for example, look for *entity* metaphors.

Examples Figure 5.7 shows how ontological metaphors are used for referral. In this figure the DATA IS AN OBJECT metaphor implicit in representing data as a document allows the user and interface to refer to specific collections of data as documents, and thus distinguish them from other data.

In figure 5.8, the ontological metaphors DATA IS AN OBJECT or DATA IS A SUBSTANCE are also used to enable quantification of a system concept. The underlying nature of the system

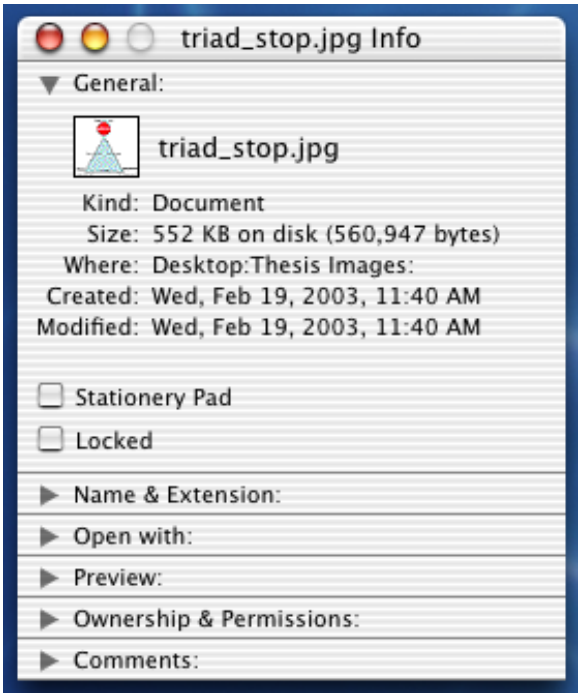


Figure 5.8: An example of quantification via an object metaphor.

concerns bits and bytes of data, but by using a metaphor that explains this data in terms of an object or substance, quantification becomes possible.

Figure 5.9 shows an ontological metaphor used to allow a causal view in the system. The metaphor THE PROGRAM IS AN ENTITY is embodied through the language used in the dialog box. This allows the user to think of the program as a physical being which is *causing* effects on their computer. Because humans are used to thinking in terms of causation in this sort of way, the program becomes more understandable, and possibly less frightening than it would be as an abstract concept.

Known uses Error messages. System "objects." Folders.

5.2.9 Structural

A structural user-interface metaphor is one which identifies a system concept with a real world concept in order to help explain it.

Relevant Sections Structural metaphors are discussed explicitly in section 3.4. They are also implicitly discussed in chapter 2 because they are what is generally meant by researchers when referring to a user-interface metaphor.

Description Structural metaphors identify a system concept with a real world concept or object in order to explain the structure of the system concept. The vehicle (the explaining concept) is generally a fairly detailed real-world concept such as "war" or "fire-truck," as opposed to the abstract nature of orientational and ontological metaphor vehicles. Structural metaphors are what has traditionally meant by HCI researchers when they discuss user-interface metaphors.

Figure 5.9: An example of causation being explained via an entity metaphor.

Figure 5.10: The trashcan in MacOS X, a structural metaphor.

Rationale The standard rationale for structural metaphor is that the user can leverage their knowledge of the structure of the vehicle to gain understanding of the structure of the tenor or system concept. This is intended to be a kind of "free" knowledge, without actual learning having to take place. Naturally, as discussed in chapter 2, it is significantly more complex than this.

Issues The key issue with structural metaphors is the *partial* nature of the structuring. As acknowledged by anyone who deals with metaphor, the transfer of structure is not a *complete* one. Therefore, there are facts about the structure of the vehicle that do *not* apply to the tenor. This clearly can lead to user confusion.

Selecting an appropriate vehicle is also a key issue when considering structural metaphor. Because the possibility of selection is vast, it is fairly easy to choose a poor candidate, or to be overwhelmed by the decision process.

Identification In order to identify structural metaphors, look for real world objects and concepts in the computer system. In many cases they will not refer to the *actual* object in the world, but in fact to an underlying system concept. Structural metaphors should be the easiest to identify because they lend themselves to overt representation in the interface as language, graphics, sound, and so forth.

Examples Examples of structural metaphor abound in modern user-interface. For two specific examples, consider figures 5.10 and 5.11. In the first, the concept of a trashcan is used in MacOS X to explain file deletion in the metaphor FILE DELETION IS USING A TRASHCAN. In the second, detailed control of music is provided by presenting it as a real

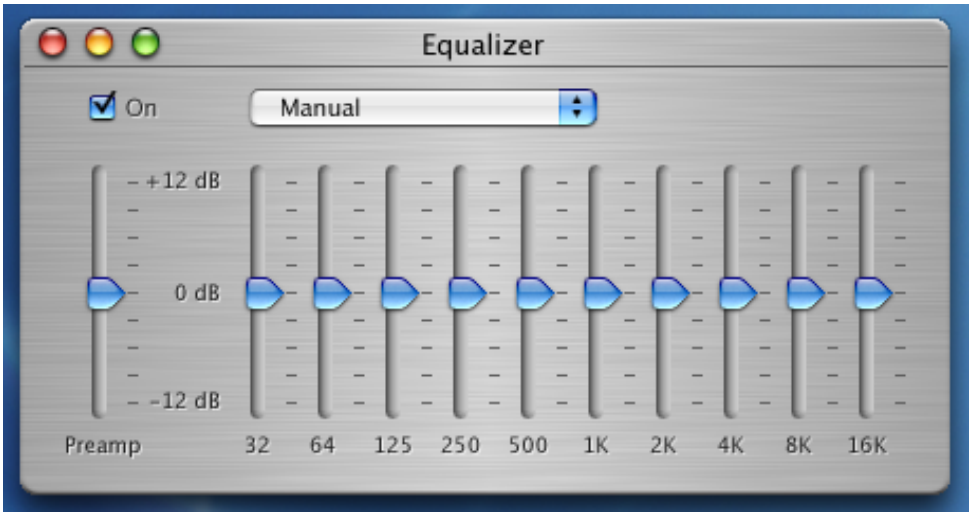


Figure 5.11: The iTunes equaliser in MacOS X, a structural metaphor.

world equaliser. Hence, the metaphor used is CONTROLLING THE VOLUME OF DIFFERENT FREQUENCIES IS USING AN EQUALISER.

Known Uses Trashcan. Desktop. Music-players. Budgeting programs. Documents. Folders. Wizards.

5.2.10 New and Conventional

A conventional user-interface metaphor is one which is already well understood by users. A new user-interface metaphor is one that is not conventional.

Relevant Sections The new versus conventional distinction among metaphors is discussed in depth in section 3.7.

Description Conventional metaphors are those which are well established in the minds of the users of a computer system. Because of this, users know what to expect when they encounter a conventional metaphor. The set of metaphorical entailments is fairly well established, and thus misunderstandings should not be a major factor.

New metaphors are the opposite of conventional metaphors. They are those metaphors that have not been previously encountered by users, or have not yet become familiar to users. Because of this, new metaphors are not well understood and users will not be sure which metaphorical entailments are applicable, and which are not.

Naturally, this is not a completely binary classification, and is rather a scale of novelty. User-interface metaphors travel on this scale from being new and not well understood, towards conventionality. Arguably, when metaphors become completely conventional, they cease to be really metaphoric at all, but simply become the way things are done.

Rationale The general rationale involved with novelty is obviously that conventional metaphors are preferable where possible. In this way the user will not have to learn a new metaphor in order to use the system. This will not always be possible, of course, and so new metaphors must be used. Note that this is also a desirable situation in its way, because there are certainly new metaphors that provide useful explanations which would be missed by the sole use of conventional metaphors.

Figure 5.12: The "traffic light" window controls in MacOS X.

The categories of new and conventional provide designers a useful way to gauge how difficult the metaphors they use will be to adopt. Newness and conventionality provide a rough sort of metric for measuring the kind of response users may have to a metaphor. They also indicate how much work will be need to be done to clarify the entailments of the metaphors, as well as in what way.

Issues Conventional metaphors and new metaphors each have their individual problems, directly related to their benefits. Conventional metaphors are already well understood by users, and this means that there are certain expectations held when a user encounters one. If a designer does not use *all* of the metaphorical entailments expected of a conventional metaphor, or uses some that are not part of the convention, user confusion could easily result. Conversely, new metaphors are problematic simply because the user does not know how the metaphor "works," they do not know what set of metaphorical entailments are valid.

Both of these problems relate to establishing the relevant entailments of a user-interface metaphor. The key to overcoming them, then, is indicating as clearly as possible in the system which implications do apply. Additionally, some expectation of users exploring the system and discovering these things for themselves seems reasonable [15]. There may also be situations where metaphor is used in a system intended for heavy or expert use. In this case the possibility of explicit training in the entailments of the metaphor becomes a possibility.

Heuristics 5 and 4 in chapter 7 provide some advice on the resolution of these issues.

Identification Identifying the novelty of a user-interface metaphor is not an easy task. Chiefly, this is because the novelty of a metaphor is relative to a *group* of people using the metaphor. Identification, therefore, is centred firmly on a good knowledge of the target users of the software. A detailed profile of the system's users will enable solid judgments to be made on which metaphors should be considered conventional and which should be considered new. Additionally, explicit surveying of a sample of target users would be a useful means with which to establish metaphor novelty.

As already noted, there is not necessarily a hard and fast distinction between new and conventional metaphors, but more of a continuum. This means metaphors could possibly be given a "novelty rating" on a scale, rather than making a binary decision as to the type.

Examples As an example of a new metaphor, consider the window control buttons in MacOS X, shown in figure 5.12. These buttons use the metaphor THE WINDOW CONTROLS ARE A TRAFFIC LIGHT to explain window functions in terms of traffic lights and how they control cars. The details of this metaphor are discussed in the previous chapter, and it is clear that this metaphor is a new one. The metaphorical entailments are not necessarily well known or accepted by its users. It is an important issue, therefore, to establish the entailments to the users clearly. Apple goes some way toward this by including symbols which appear when the user move their mouse over the window controls, as shown in the figure.

One of the most conventional metaphors in modern day user-interfaces is the "file" metaphor (THE COLLECTION OF DATA IS A FILE). The various metaphorical entailments

of this user-interface metaphor are well-established and users generally know exactly what to expect, and, equally importantly, what not to expect when interacting with the metaphor. Hence, users expect a file to contain data, but not to be written out on paper.

5.2.11 Metonymy

A user-interface metonymy is a device which represents a user-interface metaphor or metaphorical action through a related thing.

Relevant Sections The concept of metonymy is explored at length in section 3.8.

Description Metonymy represents one entity by another, related entity. In the context of the user-interface metonymies tend to be used to represent user-interface *metaphors* and the actions associated with them. Thus, a metonymy represents an interface metaphor by depicting something which is related to it.

Metonymies are especially used to represent *actions* in a system, which are otherwise difficult to visually show. As discussed in section 3.8, the relationships which metonymy uses are various. In particular metonymies such as THE PART FOR THE WHOLE and THE EFFECT FOR THE CAUSE are popular in user interfaces, as will be demonstrated in the examples section below.

Rationale As already mentioned, metonymy is used generally as a technique for representing actions in a system. Because software is very much functionality focused, this leads to considerable use of the device. An action is hard to depict literally without some sort of animation effect and so designers often use metonymy as a more simple means of representation. Thus, instead of the animation of a "new document being created," which is difficult to imagine, a metonymical solution is to depict the new document itself: the result of the action. This is an example of THE EFFECT FOR THE CAUSE or, more specifically, THE NEW DOCUMENT FOR THE ACTION OF CREATING A NEW DOCUMENT. This ability to represent metaphoric actions in an interface is what makes metonymy especially useful.

Issues User-interface metonymy's key issue involves the selection of the related entity (effectively the vehicle of the metonymy). To begin with, a particular form of metonymy such as THE PART FOR THE WHOLE must be selected. After this, a more specific version of the metonymy must be created, and finally the actual representation must be made. There are issues at each step in this process.

Choosing the base form of metonymy relates directly to the purpose the device will serve in the interface. For indication of actions, causal metonymies such as THE EFFECT FOR THE CAUSE or THE PRODUCER FOR THE PRODUCT are likely to be best. For indication of system objects, metonymies like THE PART FOR THE WHOLE are used.

Making a metonymy more specific involves choosing some particular aspect of the concept to be represented. It is difficult to suggest structured guidance for this sort of task, but it is clearly very important. It pays to remember that the selection of the representing aspect of a metonymy will be interpreted as emphasising the importance of that aspect. Thus, the metonymy THE NEW DOCUMENT FOR THE ACTION OF CREATING A NEW DOCUMENT, as shown in figure 5.13, emphasises that it is the end result of a document which is important, rather than the process of creating it. This is a good choice because the user is not interested in the underlying process of generating a new document in the computer system, but only in being able to work with that new document when it is generated.

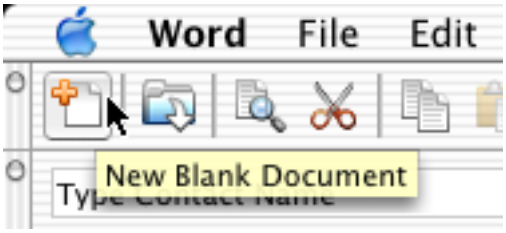


Figure 5.13: The icon displaying a blank sheet of paper for the action of creating a new document in Microsoft Word.

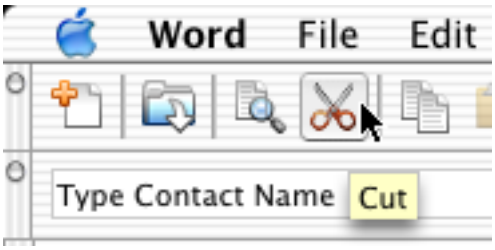


Figure 5.14: The icon displaying a pair of scissors to represent the "cut" function in Microsoft Word.

Finally, the actual depiction of the representing entity can involve problems. These issues concern graphic design, more than any more philosophical decisions, however, and guidance for designing graphics in user-interfaces can be found elsewhere (see, for example, [52]).

Identification Metonymies can be identified by focusing on the basic forms of metonymy commonly used in user-interface design. To that end, the following are typical user-interface metonymies:

1. THE PART FOR THE WHOLE is generally used to indicate metaphoric concepts of system objects. The document icons on a traditional desktop, which are a single sheet of "paper," might be considered as metonymic representations of metaphoric documents which are actually *many* pages long.

2. THE EFFECT FOR THE CAUSE is one of the popular ways of representing actions in a software system. The new document button in Microsoft Word is as excellent example, as discussed earlier.

3. THE CAUSE FOR THE EFFECT is also very useful for representing actions. It is possibly used with more frequency that the previous kind of metonymy in user-interfaces. A typical example is the print button in Microsoft Word. The depiction is of a printer which *causes* the printing of a document, which is the desired effect.

In general, user-interface metonymies will be *visual* and also will reflect real world objects more than anything else. Thus, looking for images of things from the real world will be useful for identifying possible metonymies. Once uncovered, these objects can be examined with respect to core metonymies such as those above, to see whether they fit in.

Example As an example of a typical metonymy in the user-interface, consider THE SCISSORS FOR THE ACTION OF CUTTING from Microsoft Word (figure 5.14). Here the cause of the action of cutting is used to represent the effect, and thus is a member of the overall

metonymy THE CAUSE FOR THE EFFECT. Note also that the cutting in question is actually a *metaphorical* action, rather than a literal one. The metaphorical document has text cut from one place to be "pasted" in another.

Known Uses Toolbar icons. Desktop icons.

5.3 Conclusion

This chapter has presented the material discussed in chapter 3 and 4 in a more structured and taxonomic fashion. This should allow those interested to easily access and understand the parts of the research in this thesis they are most interested in. Naturally, however, the brief approach in this chapter is not a substitute for the more detailed discussion in the earlier chapters. The chapter provides complete coverage of the internal semiotic structure of a user-interface metaphor, along with a higher level of classification of metaphors into their various types. Combined, these two views give a fairly comprehensive understanding of how user-interface metaphors function. The next chapter will demonstrate how this taxonomy can provide a useful assessment of an actual user-interface.

Chapter 6

Applying The Taxonomy

6.1 Introduction

This chapter presents two detailed case studies of the "Project Gallery" dialog which is found in Microsoft Office for MacOS X. This dialog box appears by default whenever an office program is loaded, and so is an important component of the overall interface usability. The Project Gallery dialog is displayed in figure 6.1. This chapter examines the Project Gallery using the research performed earlier in the thesis. The first case study applies the work from chapter 3, while the second case study applies the semiotic model from chapter 4 to a selection of metaphors from the first case study.

At the end of each case study some overall results will be presented to summarise the insight gained. The aim of the chapter is to demonstrate the practical application of the research in this thesis to a real user-interface.

6.2 Case Study One - Applying Lakoff and Johnson

6.2.1 Introduction

This section presents the application of the material in chapter 3 on Lakoff and Johnson's classification of metaphors. Each metaphor found in the Project Gallery dialog will be reviewed according to the following categories:

Name The title of each subsection will be the brief name of the metaphor in question. This will generally be the name of the vehicle of the metaphor (the explaining concept).

Metaphor The statement of the metaphor being used.

Representamen A description of the aspects of the Project Gallery which represent the metaphor to the user.

Description A description of the way in which the vehicle (explaining concept) generally functions in the real world compared to the purpose of the metaphor in the Project Gallery.

Classification The classification of the metaphor according to the material on Lakoff and Johnson in chapter 3.

Metaphorical Entailments A listing of *some* of the metaphorical entailments which would seem to be a part of the metaphor as identified. A complete list is not attempted because of the likely prohibitive size, instead, certain interesting entailments are pointed out.

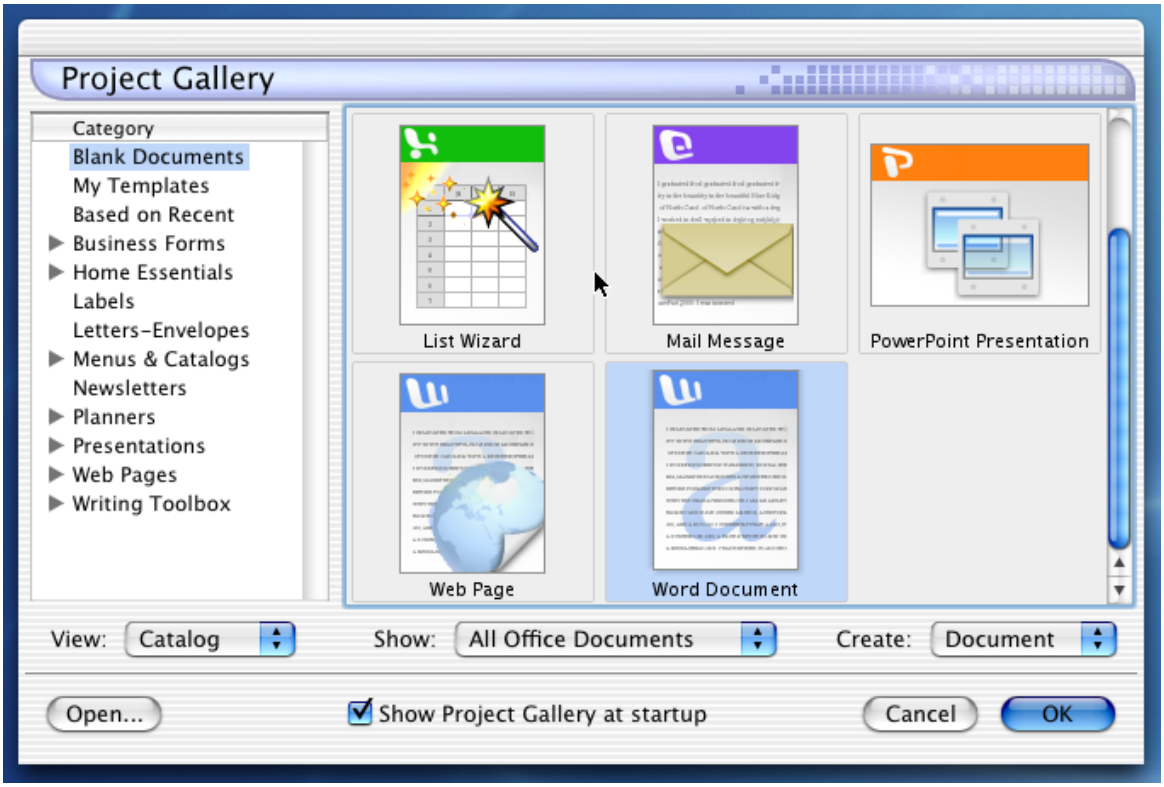


Figure 6.1: The Project Gallery window presented to the user on opening an application from Microsoft Office X.

Comments A discussion of the insight gained while identifying and classifying the metaphor.

6.2.2 Gallery

Metaphor THE COLLECTION OF TEMPLATES AND WIZARDS IS A GALLERY

Representamen

- *Language* - The title-bar of the window states the contents are a "Project *Gallery*."
- *Graphical* - The presentation of items in the display of available templates resembles the walls of a gallery. Each consists of a picture and a label, much as occurs in art galleries.
- *Entailment* - The presence of the catalog metaphor (discussed in section 6.2.3) entails a gallery metaphor by presenting a commonly associated artifact.

Description A gallery is a building which contains visual media of various kinds which are thought of as art. Art in a gallery is generally organised into one or more "shows" which contain art with some common feature (such as the artist or subject). Patrons of a gallery are invited to view the various artworks on display. A gallery will often produce a catalog to go with its shows which contains images of the works along with titles and descriptions.

The Microsoft Office Project Gallery is a collection of templates and wizards, arranged in categories (or "shows") for the user to examine and choose between. The gallery metaphor is an overarching one that appears intended to explain why the collection of information is

all available in one place. The explicit concept of a show or display is never used, but the categories available seem to serve as shows or individual galleries within the Project Gallery.

Classification

- *Structural* - A gallery is a thing in the real world that people encounter in their daily lives. It is a highly structured concept, with a large number of common features and expectations.
- *Conventional* - The gallery concept has been used for a reasonable amount of time in user-interfaces. As such, the usual metaphorical entailments should be understood by users (all that is typically meant is "viewable collection"). That said, the metaphor is not entirely conventional, and hence its entailments are not necessarily *immediately* apprehended.

Metaphorical Entailments

- the collection is a building.
- the collection contains projects.
- the collection contains things for a patron to look at.
- the collection contains valuable things that should not be touched.
- the collection is subdivided into shows which form categories of projects.
- the collection has a reception area for you to leave your bags at.
- the collection is made up of one or more large rooms.
- the collection is overseen by a director and one or more curators.
- the collection may have a cost of entry.
- the collection generally has security guards.
- the collection often has a catalog of the work it is showing.
- the collection contains valuable, unique art and artifacts.
- the collection contains items that may be for sale.
- the collection can be viewed by walking around it.
- the collection usually has white walls.

Comments It can be seen by considering and interacting with the Project Gallery window that only the most general of entailments about galleries hold. The chief entailment utilised is that a gallery is a place to store and make available for viewing works of art or, in this case, projects. Note that there is an implication here that projects are works of art.

The only other entailment that is apparent in the Project Gallery is the visual experience of a gallery, in which works of art are hung on the walls with labels. The standard view of the templates in a category, seen in figure 6.1, somewhat resembles this, with pictures and captions against a white background.

Any more detailed entailments are left out of the gallery metaphor. This may be problematic because a gallery has many entailments, and, as only a handful are used in the interface, user may expect many others they will not actually encounter. Entailments such as that the contents of a gallery are valuable and not to be touched, and that one must *purchase* the works of art to "use" them, could lead to a uneasiness in interacting with the Project Gallery.

Overall, then, the number of metaphorical entailments actually present is extremely small. This means that the gallery metaphor does not function as much more than a means to explain the overall function of the window as collecting a group of artifacts together.

An important note to make is that one metaphor can lead to another. In particular, in this interface the concept of a catalog is entailed. This entailment is actually implemented out by naming the standard view of the templates in the Project Gallery the "catalog" view, for example. As already mentioned, the notion of shows or rooms being a part of a gallery seems to be implicitly used through the categorisation of the different types of projects.

A container metaphor underlies the gallery metaphor. In particular, a gallery is a building which *contains* art-works. This container aspect is reflected in the language used, such as that "the Project Gallery *contains* templates"[1] and also in the function of the Project Gallery. The presence of the container metaphor shows how ontological metaphors can easily be present in subtle ways. In particular, as discussed in section 3.5, they often provide the *basis* of more detailed structural metaphors.

6.2.3 Catalog

Metaphor THE VIEW OF THE COLLECTION OF PROJECTS IS A CATALOG

Representamen

- *Language* - In the drop-down menu at the bottom left of the window for selecting "Views" one of the options is "Catalog."
- *Graphical* - The visual format of the viewing pane in the window is suggestive of a catalog. Each template has a picture and some descriptive text.

Description A catalog from a gallery is traditionally a book containing information on the show, or shows, that are, or have been, on display at that gallery. In general, a catalog contains pictures of the things on display along with a title and often some description and a price.

In the Project Gallery Window the catalog is a metaphor used to provide a means of displaying the various projects in a particular format. The catalog is used as one of two possible "View" options (the other being a standard list). Thus, when this view is selected, the visual format of the available projects is presented as in a catalog: each project type has an image associated with it and also a title listed beneath it.

Classification

- *Structural* - A catalog is a thing from the world that has a physical reality and specific uses. The more specific concept of an *art* catalog, which is entailed by the gallery metaphor, is even more detailed in its structure.
- *Conventional* - The catalog concept is often used in various situations in user-interfaces. The version encountered here is somewhat new because of the strong implication that it is specifically an *art* catalog.

[1] Emphasis added by author.

Metaphorical Entailments

- the view is made of paper.
- the view is a book.
- the view contains images of the projects along with labels.
- the view contains information on the projects.
- the view contains an introductory essay about the projects in the catalog.
- the view usually describes just one show.
- the view can contain prices for the items listed.

Comments The key entailment of a catalog, that it is a listing of works in a gallery containing pictures and labels, is the only one utilised. The other possible entailments are omitted, and have no representation in the interface.

One important entailment of a catalog is that it generally has more description of the art inside it than just a name. This entailment is not carried over to the catalog metaphor, and this could break with user expectation of the kind of information they will receive. Note that the "list" view of the Project Gallery contains *more* information than the catalog view. This seems to be the wrong way around.

Overall, the interface fairly clearly indicates that the catalog metaphor is to be treated simply as a means of *visually formatting* the projects in a easy to understand manner. Given that the catalog option is explicitly presented as a possible "view" of the projects, users may not expect much more than the graphical formatting constraints the catalog metaphor affords.

6.2.4 Toolbox

Metaphor THE COLLECTION OF TEMPLATES IS A TOOLBOX

Representamen

- *Language* - One of the categories available in the Project Gallery is explicitly labeled as the "Writing *Toolbox*."

Description A toolbox is a container for different kinds of tools. They are used by various professions that require a collection of tools with them at all times such as carpenters, mechanics, and plumbers. They are often of robust construction, generally metal. A toolbox is also generally built to categorise tools and keep certain things in certain places so as to easily recall where they are.

The Writing Toolbox in the Project Gallery is a container for templates to aid the user in writing particular kinds of documents. Specifically, the templates are cast as "tools" which assist in the performance of a specialised task (see section 6.2.5 for further discussion of the tool metaphor). Some of the particular templates enable the writing of a "Family Journal" or a "Book Report," for example. The templates are stored in their own categories, according to what they are a tool for.

Classification

- *Structural* - A toolbox is a real world object that has a particular function and appearance.
- *Conventional* - The toolbox metaphor or metaphors like it, such as "toolkit," have been used in interfaces for some time.

Metaphorical Entailments

- the collection of templates has compartments for the categorisation of tools.
- the collection of templates contains tools.
- the collection of templates is usually made of metal, wood or plastic.
- the collection of templates has a handle for carrying it.
- the collection of templates is portable.

Comments The toolbox metaphor uses none of the visual entailments of a real world toolbox. This is because the representamen is restricted to a statement in language of the metaphor. Any other representation of the metaphor is therefore done via the functionality involved.

It is worth noting that the toolbox metaphor is a structural extension of the ontological container metaphor, much like the gallery metaphor. The "box" component of the name explicitly indicates this. The underlying container metaphor gives the structural metaphor many of the properties that are desired for the user-interface. Specifically, the property of *storing* some kind of substance is provided, and also the ability to remove and replace items at will. The structural metaphor of a toolbox then adds entailments concerning the kinds of things stored (tools).

The key function of a toolbox that is used to extend the container metaphor is its ability to store together in one place tools that its owner finds useful. In the context of the Project Gallery these tools are the means to produce particular kinds of documents and they are organised by special categories, much as tools are organised according to function (hammers, screwdrivers, etc). Additionally, the toolbox metaphor reflects the entailment of real toolboxes that a *variety* of different of tools are stored in a single toolbox.

6.2.5 Tool

Metaphor THE TEMPLATE IS A TOOL

Representamen

- *Language* - the term "tool" appears in the label of the "Writing *Tool*box" category of templates.
- *Entailment* - the tool metaphor is directly entailed by the toolbox metaphor.

Description A tool is something in the world which is used to assist its user in achieving some particular purpose. Generally, tools have a specialised function although they can also be more broadly used.

In the context of the Project Gallery, the term "tool" is used to refer to templates that are used to produce specialised kinds of documents. They are thus said to be "Writing Tools," tools which assist in writing.

Classification

- *Structural* - If one takes a very basic definition of tool such as "useful object," then it is arguable that the tool metaphor is ontological. Given that the most common interpretation of a tool is that it is something for achieving manual work such as plumbing, carpentry and so on, however, the structural classification is the better one. Additionally, the already discussed toolbox metaphor reinforces this view of tools.
- *Conventional* - The tool metaphor is highly conventionalised. Thus, the term "tool" is used to refer to all sorts of things that the word was not originally meant for. In particular, theoretical entities such as algebra and programming environments are often described as tools, because they serve a specialised function.

Metaphorical Entailments

- the template is designed to assist in achieving a specialised task.
- the template is best used by an experienced user.
- the template can be stored in a toolbox.

Comments As already noted, the word "tool" is applied to many things other than actual, physical tools, such as programming tools (compilers, programming environments, and so on), and theoretical tools (algebra, calculus, etc). However, as already noted, the presence of the toolbox metaphor indicates that this tool metaphor involves more physical tools on the level of screwdrivers.

The templates in the toolbox are therefore implied to be like the physical tools used in the world such as screwdrivers and drills. They can be stored in a box, and they are wielded to achieve specialised tasks.

6.2.6 Template

Metaphor THE PRE-FORMATTED DOCUMENT IS A TEMPLATE

Representamen

- *Language* - The term "template" is used throughout the Project Gallery to refer to particular items. Examples include "My *Templates*," which is a category available in the Project Gallery. In the explanation of the Project Gallery, visible when a supercategory is selected, the following text is found: "The Project Gallery contains *templates* you can use for your home, business, and school projects."

Description A template is a tool used as a guide for making something accurately and consistently. Often a template takes the form of a sheet of metal which guides the user to cut material or draw on paper in a certain way. Templates allow their users to consistently create the same thing in the same way.

In the context of the Project Gallery, a template is used to provide an advanced starting point for users creating documents. Thus the template for a "Family Journal" already has various tables for listing family members, as well as appropriate images and entry types. All this allows the user to focus more closely on the content, rather than the format of the document. The templates can be used multiple times with similar results, because they always provide the same guidance.

Classification

- *Structural* - Templates exist in the real world as physical objects. Like the concept of a tool, however, templates could be considered abstract enough to classify as ontological.
- *Conventional* - The concept of a template in the user-interface has become highly conventionalised, similarly to the tool metaphor. Despite this, it was originally a metaphor, but perhaps one that has become so conventionalised by usage that it has become definitive, rather than metaphorical.

Metaphorical Entailments

- the pre-formatted document is made of thin metal.
- the pre-formatted document is used to assist its user in consistently producing the same kind of object.
- the pre-formatted document eliminates some concerns about how to create a particular kind of object.
- the pre-formatted document can be reused.

Comments The template metaphor is a fundamental when it comes to modern word-processors. There are no overt representations of the concept other than its use in language, and its comparable function to real-world templates.

Much as with real templates, the template metaphor is used in word-processors to make certain formatting decisions for the user. In theory, this allows the user to focus more closely on writing content, rather than on its format. As concerns the functionality of real world templates, then, the template metaphor is quite accurate.

6.2.7 Dialog

Metaphor THE INTERACTION PROCESS IS A DIALOG

Representamen

- *Convention* - This is a conventional way of referring to the window in which certain actions indicating preferences and decisions are made. In particular, such a window is called a "dialog box." This metaphor is not explicitly represented in the Project Gallery interface, but the Project Gallery itself *is* a dialog box.

Description A dialog is a conversation between two people. It involves the communication of pertinent information, generally in an effort to resolve some issue or make important things known. There is therefore an implication of negotiation or information exchange involved. Most important is that a dialog is communication between *two* agents, and is *bidirectional*: both participates speak and listen in turn.

In the context of the user-interface a dialog takes place between the user and the computer. Traditionally, a dialog in this sense involves the user communicating their preferences to the computer, and the computer communicating information, including which preferences are available, to the user. Note that the term "dialog" is really one specifically used by user-interface designers, rather than by the users. Therefore, it is quite possible that users will not know that the Project Gallery is a "dialog box." Because of this, insight into the metaphor discovered here will be more indicative of the sorts of things designers should consider about their intentions, rather than directly addressing user interpretation.

Classification

- *Structural* - The concept of a dialog is well defined in the world of human communication. It involves specific actions and processes.
- *Conventional* - The dialog metaphor is extremely common in user-interfaces, to the unfortunate point, perhaps, where designers do not identify it with the real world concept.

Metaphorical Entailments

- the interaction process takes place between two people with their own viewpoints.
- the interaction process involves taking turns to speak.
- the interaction process involves the communication of information between its participants.
- the interaction process involves negotiation.
- the interaction process is a social phenomenon with social rules.
- the interaction process involves the concept of politeness.
- the interaction process is generally voluntary.

Comments Because "dialog" is an abstract concept, it is difficult to *directly* represent it. One option is to present it in the form of a *script*, with indications of who is speaking at any given time. Other diagrams of the dialog process are also no doubt possible, but a satisfactory representation in the user-interface is difficult.

The concept of "dialog" describes a particular process with many special features. This being the case, it is difficult to indicate which traditional aspects of dialog are applicable in the interface, and this can be an issue. In particular, concepts of social politeness and so forth are *not* generally observed by the dialog box:

- It interrupts the user in a way which is akin to standing in front of someone and starting to talk to them without necessarily having been invited to.
- It does not use the usual forms of social politeness, such as greeting the user, saying goodbye, or any pleases or thankyous. The closest a computer dialog tends to come to this is the ability to indicate that something is "okay" to the computer by clicking a button.
- It generally does not allow an actual *dialog*, but only presents what is possible, and, in effect, commands the user to choose. There is no way to change the subject or suggest new topics for discussion.

While it would clearly be difficult to amend all of these issues, it remains that the true spirit of a dialog is not really preserved by the dialog box and more could be done. A dialog is the free communication of two agents with their own viewpoints under social conventions. The computer dialog does not allow free exchange of information, but instead a limited one, and also ignores most rules of social politeness.

Despite these issues, the concept of a dialog box is quite conventional and thus not particularly problematic. The lack of coverage of metaphorical entailments therefore indicates that there are many *improvements* which could be made by more closely following the dialog metaphor.

6.2.8 Box

Metaphor THE DELIMITED AREA ON THE SCREEN IS A BOX

Representamen

- *Graphical* - The square nature of the Project Gallery window reflects similarities with the visual nature of a box, especially a drawn one.
- *Language* - Such a window is commonly referred to as a "dialog *box*."

Description A box is a hollow object which is used to store substances and objects in in order to group and protect them. It typically has four sides extending upward from a base, and sometimes a lid or cover of some sort. Additionally, the term "box" can refer to a square or rectangle which is drawn to contain a particular area of interest.

In the context of the user-interface, the box is a means of delimiting a particular area on the screen. In this case, the box describes a window on the screen. Most important is the fact that boxes are used to *contain* things, and in this case what is contained is the information which defines a dialog. The Project Gallery window is this kind of box, and contains information on the kinds of projects the user might wish to undertake.

Classification

- *Structural* - This is a very abstract structural metaphor and is only one step away from the ontological *container* metaphor. Nonetheless, boxes do exist in the real world and one can create a standard mental image of one, perhaps the average cardboard box or a box drawn on paper.
- *Conventional* - As part of the concept of a dialog box, the box metaphor is extremely common in user-interfaces and is accepted without question by experienced users. Note that, like the dialog metaphor, the box metaphor is quite closely tied to the designer's view of the interface, although concepts such as a "check-*box*" are directly presented to the user.

Metaphorical Entailments

- the delimited area serves as a container for some substance.
- the delimited area usually has four sides and a base.
- the delimited area can be full or empty.
- the delimited area is often made of cardboard or wood.

Comments When considering how abstractly the box concept is used in the user-interface, it becomes clear that it is really just used as a synonym for *container*. No effort is really made to distinguish the box metaphor from a ontological container metaphor, other than the fact that the window representing the box is rectangular. To that extent, then, this metaphor is really just the container metaphor.

6.2.9 Document

Metaphor THE COLLECTION OF DATA IS A DOCUMENT

Representamen

- *Graphical* - Images of documents are used to represent the different projects available in the Catalog view. A typical icon is used, with a rolled up bottom-right corner and white background colour to suggest paper. There is additionally the suggestion of writing on the paper in the icon.
- *Language* - Objects representing projects in the gallery are specifically referred to as documents. Such as a "Word *Document*" in the "Blank *Documents*" category.

Description A document is most often considered to be written information presented on one or more sheets of paper. An alternative definition is that a document is any kind of recorded information such as a photo. Documents are most often created by people to communicate information to other people, and the standard process is to write them in ink either with a pen or a typewriter or, these days, with a computer. Additionally, documents are most commonly related to writing done for *work*, generally in an office environment.

In the Project Gallery, a document serves similar purposes to the above, but naturally lacks the physical nature. Nonetheless, attempts are made with user-interface documents to imitate the physical nature of documents as far as possible, to reinforce the metaphor. Thus, although there is no real paper in a computer, computer documents are represented in the interface *as if* they were on paper, as with the traditional icons for documents with folded down corners.

Classification

- *Structural* - Documents are real world objects that have quite particular properties, such as being made of paper, written on with ink, and so on.
- *Conventional* - The computer document is such a conventional metaphor that many people will have trouble recognising it as such. This may be another case, like the template metaphor, of an interface metaphor becoming *definitive*.

Metaphorical Entailments

- the collection of data is made of paper.
- the collection of data contains writing.
- the collection of data can be filed in folders.
- the collection of data can be mailed to others.
- the collection of data is most common in an office environment.
- the collection of data is produced on a type-writer or with a pen or pencil.
- the collection of data can be thrown into the trashcan.
- the collection of data can give paper-cuts.
- the collection of data has a quantifiable size.
- the collection of data has a location in the world.

Comments The document metaphor, which is *central* to the use of today's desktop computers, clearly utilises a great many of the metaphorical entailments discussed above. This is the first metaphor encountered in the case study which uses an overt graphical representation of the metaphor, and one of the few that utilises a large number of metaphorical entailments. Additionally, the document metaphor has an object-like status in the interface, which introduces at least some of the physical nature of documents to the computer.

The document metaphor is tied in to the general workings of today's computer systems, and works together with the trashcan, file-folder and desktop metaphors (to name three) in contributing to the overall office metaphor which is discussed in section 6.2.11.

It is important to note that computer documents have features over and above the metaphorical entailments available from real world documents. Thus it is far easier to edit a metaphorical document than a real one, as well as easier to make copies and so forth. This is the "magic" that Alan Kay speaks of as being the most important aspect of user-interfaces; the parts *not* indicated by the metaphor [40]. It is equally important to see, though, that without the underlying metaphor, the extending, "magical" features would not have been possible. The metaphor makes the magic possible.

6.2.10 Wizard

Metaphor THE DIALOG BOX(ES) IS A WIZARD

Representamen

- *Graphical* - The icons representing wizards have a wand as a part of the image which indicates the concept of magic and wizardry.
- *Language* - One of the items available in the Project Gallery is labeled as the "Envelope *Wizard*," another is the "List *Wizard*."

Description A wizard is a person who performs magic. Traditionally, wizards are extremely powerful individuals with somewhat mysterious motives who wear long gowns and have long beards. Wizards are often viewed as being quite cryptic in Western culture, and can just as easily be evil as good. Additionally, wizards are traditionally *male*, and are contrasted with *witches*. Interestingly, witches are almost always evil, while wizards are not necessarily so.

In the user-interface a wizard is a tool that aids the user in achieving some, otherwise complex, task. It does this by guiding the user through the process step by step, generally with helpful information along the way. Importantly, the wizard performs the user's task with the information provided, shielding the user from the actual process.

Classification

- *Structural* - A wizard is a detailed concept real world with many particular entailments.
- *Conventional* - The wizard metaphor has been in use for some time and is accepted by most experienced users. Some new users may still be unclear on the exact entailments of the metaphor, however.

Metaphorical Entailments

- the dialog box is an object.

- the dialog box is an entity.
- the dialog box is a person.
- the dialog box performs magic.
- the dialog box can be helpful or a hindrance.
- the dialog box has a beard.
- the dialog box has a wand.
- the dialog box is very powerful and is not to be trifled with.
- the dialog box is extremely knowledgeable.

Comments As can be seen from the suggested subset of entailments above, a very large number of the metaphorical entailments are not used at all in the user-interface. In general, user-interface wizards have little to do with the wizards of the real or fantasy world. In fact, so few of the entailments are used, it becomes difficult to genuinely claim the metaphor is being used at all.

The only entailments genuinely utilised are the suggestion a wizard has a wand, and that the wizard is very knowledgeable and may be helpful. All other entailments have been jettisoned. Most importantly, the entailment that the wizard is a person entails many social rules that should be obeyed during any interaction. The rules of social interaction tend to be ignored when dealing with wizards in the user-interface, although they are sometimes depicted as smiling or appearing friendly. This lack of social convention is a problem in user-interfaces at large, as already suggested earlier, and also as discussed in *The Media Equation* by Byron Reeves and Clifford Nass [70].

Additionally, it is clear that many of the common entailments about wizards are inappropriate in the user-interface. This is especially true of entailments that suggest wizards might be evil (which is countered by having the wand emit "white magic") and that the wizard can be cryptic, both of which are completely undesirable in a user-interface intended to *aid* the user to complete some task.

The most important entailment used in the wizard metaphor is that wizards are knowledgeable and can use magic to achieve tasks that the user would find difficult or overly complex. This, however, does not completely distinguish the metaphor from others such as an expert metaphor, or a post office official metaphor, when it comes to creating formatted envelopes, for example. While the wizard metaphor is interesting, it leaves far too many metaphorical entailments unused, especially those concerning social interaction, and brings into question its utility as an explanatory device.

6.2.11 Office

Metaphor THE COLLECTION OF SOFTWARE IS AN OFFICE

Representamen

- *Language* - In the "Show" drop-down menu positioned at the bottom centre of the Project Gallery the default option is "All *Office* Documents."
- *Entailment* - The office metaphor is also entailed by other metaphors which stem from it. There is a "typewriter" for writing office documents (Microsoft Word), a program

for doing figures and tabulations (Microsoft Excel), a program for preparing presentations (Microsoft PowerPoint), and a program for taking care of office correspondence (Microsoft Entourage). The presence of these aspects of day-to-day office life help to represent the office metaphor functionally. The metaphor is enforced further by the fact it matches well with the desktop metaphor which includes further metaphors such as the trashcan, documents and the desktop. This overall collection of metaphors helps to suggest an office.

Description An office is an environment where people do their jobs. The traditional view of an office involves certain tools and activities which are common to a broad range of offices. Thus, it is extremely common to find some means of preparing documents such as a typewriter (though it is almost always a computer these days), a way of calculating numbers such as expenses or budgets, a means of corresponding with other offices, and materials for preparing presentations. Although all of these things tend to be performed on computers in today's offices, they all had predecessors such as typewriters, ledger books, overhead transparencies, envelopes, and paper.

The office metaphor is based on the concept of the "paperless office." Thus, the metaphor suggests that the software is "the same thing as an office, only without the paper." All the tasks that were normally performed on paper are now performed on the computer. Functionally, they are very similar, and this is why the metaphor functions well, as will be shown.

Classification

- *Structural* - An office is a particular thing in the world with specific entailments.
- *Conventional* - The office metaphor has been around for a reasonable amount of time (Microsoft Office was launched in 1990) and is generally accepted by its users. The contents of office software has been standardised to a degree that users tend to know the sorts of things to expect.

Metaphorical Entailments

- the computer software has a desk in it.
- the computer software has a typewriter in it.
- the computer software has envelopes and mailing material in it.
- the computer software has presentations materials in it.
- the computer software has a ledger in it.
- the computer software has a filing cabinet in it.
- the computer software has documents in it.
- the computer software is for doing work in.
- the computer software has a trashcan.
- you can have your lunch in the computer software on a busy day.

Comments The office metaphor is one of the overarching metaphors in today's computer systems. It combines with the desktop metaphor to create an overall coherence which structures the user's experience of the software. Effectively it extends the desktop metaphor to add actual tools for performing work, rather than only managing documents in a kind of virtual filing cabinet. Because the desktop metaphor came first, however, the office metaphor is strangely *contained* by it. This is despite the fact that, in the real world, an office contains a desk, and its desktop. Effectively, this occurs because the desktop metaphor is really an office metaphor in disguise. People do not really keep all their filing on top of their desk, nor do they place their trashcan on top of it. The parts of the desktop metaphor more accurately reflect the way in which an office is used to store and retrieve information. The office metaphor of Microsoft Office, therefore, *extends* this metaphor by adding in the common "tools of the trade" that are found in an office. The Microsoft Office metaphor is a *container* for software which achieves work-based tasks.

6.2.12 Web

Metaphor THE INTERNET IS A WEB

Representamen

- *Language* - One of the options available in the "Blank Documents" section is the creation of a "*Web*page."

Description A web is a kind of latticed or woven structure. The immediate association for many people is that of a spider's web. The important point about webs is that they are generally composed of many threads joined together, often in a systematic and patterned way. They are also traditionally used for *catching* things.

In the context of the user-interface, the web refers to the structure of the Internet, and "web" has become another term used to refer to the Internet, most specifically the Internet encountered through today's browsers using hypertext. The Internet is composed of millions of sites which are joined together via "links." It is not difficult, then, to imagine this as a giant web, where the links form the thread of the web, and the sites are situated at the junctions where threads meet.

Classification

- *Structural* - A web is a real thing in the world, as discussed above.
- *Conventional* - The web metaphor has been around for a fairly long time, and the term is generally used without any thought to what webs are in the real world. In fact, the Internet may have become the primary definition of the term "web" for some people.

Metaphorical Entailments

- the Internet is woven together out of threads.
- the Internet is a highly complex, although regular, structure.
- it is possible to get from any one point on the Internet to any other by following various threads.
- the Internet contains structural redundancy.

- the Internet is used to catch things.
- the Internet was made by a spider.

Comments The key point is that a web is rarely haphazard, but is made in a particular way. This is not completely that true of the Internet, which is quite chaotic in its nature. Nonetheless, the notion of a vast interconnected set of points does fit the notion of a web quite well. Note also that the net metaphor which the basis of "*net*work" and "Inter*net* relates structurally to the web metaphor. The term "web," however, does not really describe the entire internet, but only webpages. It thus omits things such as email and newsgroups, which are also part of the overall Internet.

6.2.13 Page

Metaphor THE COLLECTION OF DATA IS A PAGE

Representamen

- *Language* - One of the options available in the "Blank Documents" section is the creation of a "Web*page*."
- *Graphical* - The icons displayed for documents also reflect the concept of a "page" because they represent a single sheet of paper, including a rolled up corner.

Description A page is a sheet of paper, most often thought of as occurring in a book. A page usually has written material on it, and may also contain images and other media. The key point is that a page contains information in a particular format: text, and possibly images, most often on a rectangular sheet of paper. The layout is always such that some particular amount of information fits on the page, which is of a certain size. Other formatting considerations include sections, titles, headings and so forth. A document is made up of individual pages, so pages are a means of constructing documents and are linked in a sequence.

In the context of a user-interface a page is, once again, a subpart of some form of document. A "webpage" is a collection of data that is a subpart of a "website." In this environment pages are joined together by "links" rather than necessarily being strictly sequential. The collection of webpages defines the website, just as the collection of pages in a book is the book itself (without covers). Webpages, just like real world pages, contain information, generally in the form of text and images. Additionally, webpages are set out in a similar manner to the pages in a document: they involve a fixed width of text, with the reader reading downward to obtain more information.

Classification

- *Structural* - A page is a real thing in the world, occurring in books, magazines, and other media.
- *Conventional* - The page metaphor is so commonly used that it generally goes unnoticed, but is instead seen as an alternative definition of "page" by many people.

Metaphorical Entailments

- the collection of data is made of paper.
- the collection of data generally has writing on it and contains information of some kind.
- the collection of data generally is part of a larger collection of pages.
- the collection of data has a particular position and purpose in the larger collection of pages.
- the collection of data has a particular kind of format.

Comments The page metaphor is quite similar to the document metaphor, but has fewer entailments concerning the context of use. A page is really just a sheet of paper with information on it, rather than something so closely associated with work as a document. Additionally, one can have a blank page, but not a blank document. Note the conflict here between this fact and the presence of the "Blank Documents" category of the Project Gallery.

The key entailment of the page metaphor is that it is normally part of an overall collection of pages that bear some relation to each other and are linked. This entailment is applicable, in that an individual webpage can be linked to other pages which are relevant. The relevance does not necessarily relate to the page being in a *sequence*, however, but generally indicates the information on the pages are related. Additionally, the traditional format of a page is represented in a webpage, which presents a block of information intended to be read from top to bottom.

6.2.14 Window

Metaphor THE RECTANGULAR AREA ON THE SCREEN IS A WINDOW

Representamen

- *Graphical* - A window generally has four sides, and a view through it. This is reflected in the computer window which is four-sided and presents a view of particular information provided by the software.
- *Language* - Users and designers refer to the "windows" on the screen. This is not explicitly stated in the Project Gallery, but it is a window, and so the metaphor is part of it.

Description A window is something used in buildings to provide a means of letting light in, and to provide a "view" of the outside world to those who are inside. Additionally, windows provide a view of the *inside* of a building to those who are outside. They are traditionally made of glass, and have a frame around them which defines the boundary of the window. Windows may sometimes be split into subparts which are called "panes."

In the context of the user-interface, a window is a rectangular area in the interface which contains a view of some particular information. In the present case study the Project Gallery window provides a view of information about the set-up of a session with Microsoft Office and offers certain options to the user. Additionally, user-interface windows can have panes, as discussed in section 6.2.15. Windows in the user-interface are able to *overlap* each other which is not a property of real windows.

Classification

- *Structural* - A window is a real thing in the world, common to almost every kind of building.
- *Conventional* - The window metaphor is a very standard one, and has existed at least since the Xerox Star [38].

Metaphorical Entailments

- the rectangular area is made of glass.
- the rectangular area is see-through.
- the rectangular area can contain separate panes.
- the rectangular area has a frame.
- the rectangular area can have a ledge.
- the rectangular area allows us to see a view.
- the rectangular area lets light into the room.
- the rectangular area is in a wall.
- the rectangular area cannot overlap with other windows.
- the rectangular area shows a unique view.

Comments The basic function of a window is that of providing a view into a different set from that offered by the rest of the screen. This allows users to divide up the information on the screen into manageable areas. This partly matches with the real world window, in that both *compartmentalise* a particular view.

The detailed physical entailments of a window are largely ignored. Instead, only basic entailments such as that the window is rectangular, contains a view, and may have separate panes are used. A window in the user-interface has many properties that conflict with real world windows, such as that a window can be resized or hidden, and can overlap with other windows. These entailments do not simply *extend* the window metaphor, but directly conflict with important entailments of it.

6.2.15 Pane

Metaphor THE SUBSECTION OF THE WINDOW IS A PANE

Representamen

- *Language* - Subsections within a window on a computer are often referred to as "panes." Knowledge of this metaphor may be more restricted to user-interface designers than actual users. As already noted previously, however, this simply means that insight gained here is informative for designers examining their intentions, rather than for insight into the effect on users.

Description A pane of a real world window is simply the actual glass in the window. Most interestingly, a single window can have *multiple* panes, each divided from the others via extra framing. The typical example of this is a window which has four panes dividing it into quarters.

The same concept of division is intended in user-interfaces, where the panes of a window divide the information contained, or the "view", into subsections. Thus, one pane might contain navigation information, and one might contain written content. In the Project Gallery window there is a pane for selecting the category of project the user is interested in, and another pane in which the available templates and wizards relating to that type are displayed.

Classification

- *Structural* - The pane of a window is an actual thing in the world.
- *Conventional* - Although the pane metaphor would likely be new to users who encountered it, it is never directly referenced in the interface. Therefore, it is largely those involved with user-interface design who know of this term and metaphor. The metaphor is a conventional one to designers.

Metaphorical Entailments

- the subsection of the window is made of glass.
- the subsection of the window has its own frame.
- the subsection of the window divides up the view.
- the subsection of the window presents a subpart of the overall view of the window.

Comments The pane metaphor is interesting in that it is an *entailment* of the window metaphor. It is, therefore, a good example of the fact that metaphors can entail other metaphors in the user-interface which can then also be implemented. This process enhances the overall coherence and consistency of the interface.

The entailments of a pane are largely maintained. User-interface panes do divide the view of a window, although they do so in a more functional way than in the real world. The panes in the user-interface also do have their own frames to make them visible. Generally, then, the pane metaphor is quite comprehensively implemented in the interface, despite not being a metaphor that users are necessarily aware of.

6.2.16 Mail

Metaphor THE INFORMATION TRANSFER IS MAIL

Representamen

- *Graphical* - The icon for the option of creating a "mail message" shows a document with writing on it along with an envelope. The envelope in turn refines our interpretation of the document such that we believe it to be a letter.
- *Language* - In the "Blank Documents" category is the option to create a "*Mail* Message."

Description Mail is used to communicate information between people in the real world. The general process is to write a document, place it in an envelope, add stamps, and then post it to someone else so that they can read it. The most common thing to mail is a letter, but almost anything can be mailed in reality.

On computers mail is often referred to as *email* or "electronic mail." The mail metaphor is intended to help users to understand the process of electronically sending data from one computer to another. Thus, one writes a mail message, and then sends it to a particular address on the Internet. Many features are shared between real world mail and the mail metaphor such as having senders, receivers, addresses, attached documents, and so on.

Classification

- *Structural* - Mail is a concept from the real world with a great deal of structure associated with it. This varies from artifacts involved, such as stamps and envelopes, to processes such as addressing and mailing.
- *Conventional* - Email is one of the most popular uses of computers today, and it has become a very familiar way to send information. The fact it is metaphorical and based on real world mail is likely not thought of much anymore, and the entailments are fairly well acknowledged by users.

Metaphorical Entailments

- the data transfer involves a post-box.
- the data transfer involves sending information to a particular address.
- the data transfer requires an envelope.
- the data transfer involves postal workers.
- the data transfer requires stamps.
- the data transfer has a sender and a receiver.
- the data transfer has people who serve as intermediaries between sender and receiver, taking the mail from one to the other.
- the data transfer does not guarantee arrival time of information.
- the data transfer can be unreliable.

Comments The mail metaphor has become hugely successful as a way of communicating between people using computers. A large number of the major entailments are used, which may be one reason why acceptance of the new technology has gone so well.

Although some of the entailments of real world mail are somewhat inappropriate, the huge usage of email has meant that almost all users have learnt the correct entailments, conventionalising the metaphor. Entailments such as that an envelope and stamp are required might conceivably confuse very new users, but, overall, the metaphor is so comprehensively defined that problems ought not to arise.

6.2.17 Entourage

Metaphor THE SOFTWARE IS AN ENTOURAGE

Representamen

- *Language* - One of the options in the "Show" drop-down menu is to show "*Entourage* Documents."

Description An entourage is most typically a group of people who look after the interests of some central, more important person. They enable the person to concentrate their valuable time on specific tasks while the entourage takes care of the details.

The software Microsoft Entourage is a combination of Microsoft Outlook (an email and organiser program) and the Palm desktop (for interfacing with a PDA). The software collectively allows the user to manage their day-to-day activities as well as communicate with the world through email using the mail metaphor of section 6.2.16. The software can be seen as an entourage which takes care of things such as mailing the user's documents, reminding the user when they have to do particular things, and remembering what the user's schedule is, for example.

Classification

- *Structural* - An entourage is a particular concept in the world which has a quite specific definition.
- *New* - The entourage software is relatively new, and users may not know exactly what is entailed by it. In fact, many users may not know what an "entourage" is.

Metaphorical Entailments

- the software is a group of people.
- the software looks after the tiresome details of everyday tasks.
- the software helps organise the central person's everyday activities.
- the software is always around the central person.
- the software is used by important people.
- the software sacrifices its own time for the sake of the central person.

Comments The entourage metaphor is an interesting one, in that is is so suggestive of a group of *people* who assist the user. In actual fact, there are no virtual people represented but instead a suite of programs which serve some of the same purposes as an entourage. Lost in this is the fact that interacting with an entourage is a *social* interaction. None of the social entailments appear to be implemented in the interface, which omits a significant aspect of an entourage. Perhaps the argument is that only the functional aspects of an entourage are relevant to computer software. As already noted, it is quite possible that the term "entourage" will be foreign to many users of the software, which would lead the metaphor to be lost entirely.

6.2.18 Scroll

Metaphor THE AREA OF THE WINDOW IS A SCROLL

Representamen

- *Graphical* - Although slim, the information in a window does look like the visible information on a scroll. Additionally, when progressing through the information it is moved upwards, much as in a scroll.

- *Language* - The bar alongside a window which controls its movement is referred to as a "*scroll*-bar." The action of moving through the information using the scroll-bar is called "*scrolling*."

Description A scroll is a long sheet of paper which has its ends rolled such that information can be displayed in controlled amounts. When reading new information the paper is "scrolled" upward, so that the lower parts of the paper are displayed.

In the user interface a scroll-bar is used when there is more information available than can be fit on the screen. In this case, the scroll-bar enables the user to scroll the information in and out of the viewable area, thus making it all accessible.

Classification

- *Structural* - A scroll is a real world artifact with a definite form and purpose.

- *Conventional* - The scroll-bar has been around for a very long time.

Metaphorical Entailments

- the area of the window is on paper with rolled ends.

- the area of the window is a view of part of a larger amount of information.

- the area of the window can display different parts of information by scrolling.

Comments The scroll metaphor has been around for a significant amount of time and is well understood by most users. The functionality of the metaphor reflects the real world act of using a scroll fairly accurately, and so the metaphor is a success in that sense. Other than the omission of entailments concerning the physical appearance of a scroll, such as the rolled ends of paper, the scroll metaphor stays quite true to its vehicle (explaining concept).

It is also interesting to note that one could suggest an orientational metaphor is also present, as indicated by the arrows on the scroll-bar. The metaphor here would be something like MORE INFORMATION IS DOWN. This accounts for pressing the down arrow in order to see new information (and the up arrow to see the old information). Such a metaphor extends back to the use of real scrolls, however, and is thus not unique to the user-interface.

6.3 Results of Case Study One

The just completed case study reveals some interesting general considerations about the metaphors used in the Project Gallery. Some of the more important observations will now be discussed to gain some insight into the interface's design.

Dominance of structural metaphors

One thing which is very clear is that structural metaphors are easily the most common of the three types used. Of the seventeen metaphors identified, all can be considered structural. That said, there are some borderline cases such as template, tool, and box which might be considered ontological. Additionally, a possible orientational metaphor was identified during the discussion of the scroll metaphor.

The reason for this prevalence of structural metaphors is fairly clear: structural metaphors provide the most detailed concepts with which to work in the interface. The more detail there is available, the most opportunities there are to leverage existing user knowledge. Additionally, structural metaphors are by far the easiest to indicate in the representamen, which means they have a much better chance of being identified by users.

Dominance of conventional metaphors

Conventional metaphors form the lion's share of the metaphors in the Project Gallery. While there is one genuinely new metaphor in "entourage," new metaphors are effectively not present. One reason for this is that Microsoft Office has been around for some time, and thus its metaphors have become part of the general user consciousness. Additionally, the Project Gallery is designed to help newer users easily create documents, and so it may be a design decision to limit the sorts of metaphors used to common ones. Finally, Microsoft is a large and fairly conservative company, and is unlikely to wish to surprise users overly with its interfaces. The new/conventional distinction will be discussed some more later in this section.

Entailment selection largely functional

When examining the kinds of entailments actually present in the user-interface it is apparent that the large part of them relate to the functional nature of the vehicle. Although there are some perceptual aspects of metaphors used, the largest number of entailments are functional ones. The reason for this is, presumably, that software is driven by function and so the most useful kinds of entailments are those that explain the *workings* of the software. Additionally, the use of too many graphical representations of metaphors would likely lead to a cluttered and confusing interface. Also, as already mentioned, Microsoft is more likely to be conservative in its choices of representamens than other, less established, software companies might be.

Prominence of language and graphical representamens

When perceptual aspects of a metaphor *are* used, they are biased toward the graphical, either as images or language. In fact, these are effectively the only kinds of representamens present in the metaphors discussed. The Project Gallery interface does not involve any other forms of perception such as sound or kinaesthesia. The one exception to this might be the act of *scrolling* the contents of a window, which has some ties to the real world act of reading a scroll. This bias toward graphical representamens is certainly not unusual: the chief presentation device of a computer is its screen, which leans interaction toward visual stimuli.

Entailment of other metaphors

As was seen in the case study, metaphors can entail other metaphors. In this way the gallery metaphor entailed the catalog metaphor, the window metaphor entailed the pane metaphor, and the toolbox metaphor entailed the tool metaphor. By implementing entailed metaphors

a greater coherence can be created in the interface, because the implementation of an entailed metaphor is *guaranteed* to cohere with the metaphor that implied it.

The use of entailed metaphors demonstrates the usefulness of noting as many metaphorical entailments of a potential metaphor as possible. Useful pieces of functionality can be discovered in this way. In each case, the metaphor extended the structure of the interface considerably, and contributed further functions.

Prominence of objects over concepts

Most of the metaphors in the above case study involve concrete objects or entities from the real world, rather than concepts. This suggests it is more common to have physical things from the real world as vehicles (explaining concepts), such as catalogs, rather than abstract concepts, such as beauty. Again, this is likely because of the amount of structure lent by physical objects, as well as their more directly perceivable nature.

The prevalence of physical artifacts from the real world also indicates the popularity of structural metaphors which have an ontological underpinning. Such metaphors generally build on object or entity metaphors as a basis. The presence of a basic ontological metaphor provides a considerable amount of well-known information about the vehicle in question. Additionally, agreement on the properties of real world objects is likely to be far greater than on the nature of abstract concepts. This kind of agreement is desirable, as it will help lead users to predictable approaches to the interface, which is more controllable.

Mixing metaphors

While using entailments to create new metaphors enhances coherence, using disparate metaphors helps to destroy it. This is apparent in the co-occurrence of such metaphors as wizard, gallery, and office. Those three metaphors have little or nothing in common, and so it is, in some sense, peculiar to have them all in the same interface. Such mixing may well disturb users, especially novices who are less able to accept that different metaphors can co-exist in a single interface without interfering with each other. A new user might wonder, for example, why the wizard would be spending time in a gallery, or why there is a toolbox lying around. In general, the user should be shielded from such concerns.

Cultural considerations

One interesting feature of the case study is that no serious cultural considerations immediately came up. There do not appear to be metaphors in the interface that would cause major issues based on the culture a user comes from. It is certainly true that most of the metaphors are intended in a Western culture sense, but, on the whole, the metaphors are generalisable across cultures. Concepts such as tool and catalog are fairly universal, for example. This makes sense for such a international product as Microsoft's Office.

Omitted entailments

The case study also demonstrated that it is problematic to omit certain entailments. In particular, it is an issue to omit entailments which are of central importance to the vehicle. An example of this is the mistreatment of the wizard metaphor and the social entailments of the fact a wizard is also a person. The expectation of social interaction with anything thought to have a personality and feelings is a strong one in humans, and omitting this detail may well be significantly disturbing to users.

Presence of ontological metaphors

Although there are no *solely* ontological metaphors in the Project Gallery interface, it is clear that they underly a large number of the structural metaphors. As already mentioned, most metaphors in the Project Gallery are based upon object and entity metaphors. Additionally, the container metaphor underlies metaphors such as gallery and toolbox, and the entity metaphor underlies the wizard metaphor.

What is demonstrated is that, although there were no stand-alone ontological metaphors, they did play a significant role in the shaping of the entailments in the interface. The ontological metaphors provide all of the basic entailments, allowing the overlying structural metaphor to add important details. This arrangement allows users to apply their basic knowledge of the world quite easily, before focusing on the exact nature of the structural metaphors.

Differentiating structural and ontological metaphors

A point raised by metaphors such as tool and template is that it is not always straightforward to state whether a metaphor is structural or ontological. That is, there are some borderline cases, where a metaphor could be reasonably classified as either. What this suggests is that there is something of a continuum between the ontological and structural categories. Ontological metaphors are abstract characterisations of the basic things in the world. Structural metaphors are ontological metaphors made more detailed, by specifying what *kind* of thing they are. The level of detail in the metaphor, above its basic ontological properties, therefore, would seem to define the metaphor's place on the continuum.

Differentiating new, conventional and definitive metaphors

The notion that a metaphor could become so accepted as to be an alternative definition of its vehicle was raised in examining metaphors such as toolbox, template, and web. This, along with the nature of new and conventional metaphors, suggests a complete life-cycle for user-interface metaphor. Specifically, it appears that a metaphor begins as new, with users not necessarily grasping all the UI metaphorical entailments, proceeds toward convention, as more and more users come to accept it, and eventually becomes a definition, when all people accept and understand the metaphor completely. It is presumably desirable to have new metaphors eventually become definitive as it implies they have been totally accepted by their users.

Designer only metaphors

It is suggested by the existence of user-interface metaphors such as pane and dialog that there are metaphors used only by the *designers* of the interface. Such metaphors appear to be useful to the designer in explaining functionality to *themselves*, rather than to the users. These metaphors, therefore, guide design, without overtly guiding the users of the final product. An awareness of these metaphors is especially important because they directly reflect the designer's thoughts about how the interface should work fundamentally.

6.4 Case Study Two - Applying the Semiotic Model

6.4.1 Introduction

This second case study is intended to indicate a possible use for the semiotic analysis of user-interface metaphors provided in chapter 4. Although it should be clear from the work in that

chapter that the semiotic analysis is more geared toward analysis of the design process, this case study will show it can also be used for retrospective analysis of metaphors. In particular, it makes it possible to speculate in a structured manner about the metaphor and the possible design decisions which were taken.

The basic format of this section is to provide an analysis of three metaphors from the Project Gallery interface, gallery, catalog, and wizard. Each metaphor will analysed into the parts discussed in the semiotics chapter earlier. Although the relations will not be explicitly discussed in order to keep the amount of material manageable, they will be covered in the course of examining links between the parts of the metaphors examined.

6.4.2 Gallery

Metaphor As already stated in the previous case study, the metaphor involved here is most likely:

THE COLLECTION OF TEMPLATES AND WIZARDS IS A GALLERY

The metaphor therefore uses the concept of a gallery as a means to explain the collection of certain types of data and functions available in the software.

Metaphorical Entailments As already stated several times in this thesis, it is extremely difficult to list *every* metaphorical entailment that might be possible, or, in this case, that the designer might have considered. In fact, it is more than likely that no systematic approach was taken to the design of the metaphors in the Project Gallery. Nonetheless, it is possible to speculate as to the sorts of entailments that could be expected. By listing some of these, the collection can later be compared with the UI metaphorical entailments as a means of analysis of the overall design:

- the collection is a building.
- the collection contains projects.
- the collection contains things for a patron to look at.
- the collection contains valuable things that should not be touched.
- the collection is subdivided into shows which form categories of projects.
- the collection has a reception area for you to leave your bags at.
- the collection is made up of one or more large rooms.
- the collection is overseen by a director and one or more curators.
- the collection may have a cost of entry.
- the collection generally has security guards.
- the collection often has a catalog of the work it is showing
- the collection is contains valuable, unique art and artifacts.
- the collection contains items that may be for sale.
- the collection can be viewed by walking around it.

- the collection usually has white walls.

It is notable that there are certain entailments which lead to further metaphors. One in particular is that a gallery is a *building*. Buildings have many properties of their own, which therefore fall in the scope of the gallery metaphor. A building metaphor in turn draws from the container and object metaphors, because it is an example of both ontological categories. It is important to be aware of the entailment of more basic metaphors, because each indicates a large set of entailments implicit in the overall metaphor.

Another important area of consideration is that of metaphor detail. By itself, the concept "gallery" does not define a very specific thing in the world. In particular, it does not suggest the *kind* of gallery intended. The concept, therefore, suggests a generic gallery with the very *basic* properties that all galleries tend to have such as white walls, rectangular rooms with art in them, security systems, and so forth.

The key entailments of a gallery, it can be claimed, are those relating to its function or purpose. If the tenor (concept to be explained) does not match well with the function of the vehicle (explaining concept), then the metaphor is not apt at all. A gallery is, at its most basic, a place for storing and viewing a collection of artifacts thought of as art. This does match fairly well with the function of the Project Gallery, which is to store for viewing a collection of artifacts thought of as useful for performing work. Note that there is an entailment that the templates and wizards in the gallery are *art*. This seems inappropriate, but more or less impossible to avoid, given the gallery metaphor. Art has properties such as high value and delicacy which do not match well with the idea of software tools for achieving work.

Finally, a metaphor can entail new metaphors that go along with it. As an example, consider that the gallery metaphor entails a catalog metaphor. If the overall metaphor matches the purpose of the software well, then there is some reason to believe that entailed metaphors will be useful also. That is because, if the vehicle and tenor serve similar enough purposes, then those entailments useful to the vehicle have a greater likelihood of being useful to the tenor. This is the case with the catalog metaphor, which provides an intuitive graphical layout of the templates and wizards available.

Designer's Interpretant It seems clear that the designers did not wish to use very much of the gallery metaphor in the interface. Thus, it is quite sparely represented in the representamen. The basic interpretation the designers appear to have been led to is that a gallery metaphor is useful for explaining why a collection of artifacts is all in one place, as well as stored in categories. Additionally, it is obvious that the designers thought of the entailment of a catalog and found it useful.

Representamen The representamen for the gallery metaphor rests largely on the use of language. The window is named and referred to as the "Project *Gallery*." Additionally, the presence of the catalog metaphor, also most obviously indicated by language, reinforces the concept of a gallery in the interface. This is a case of a sub-metaphor forming part of the representamen for its super-metaphor.

The rest of the representamen is reflected in the interactive properties of the window. That is, the fact that the window *displays* a collection of artifacts for use by the user indicates that the gallery metaphor applies.

UI Metaphorical Entailments It is possible to examine the representamen as discussed above and to deduce from this the likely UI metaphorical entailments. Note also that, given a

formal development process was probably not used, there likely were no specific UI metaphorical entailments in mind when the metaphor was developed. Nonetheless, some of the UI metaphorical entailments appear to be:

- the collection contains projects.
- the collection contains things for a patron to look at.
- the collection is subdivided into shows which form categories of projects.
- the collection is overseen by a director and one or more curators.
- the collection often has a catalog of the work it is showing.
- the collection usually has white walls.

An immediately apparent feature of the above entailments is that they largely concern the *function* of a gallery, rather than its perceivable properties. This was suggested by the lack of a perceivable representamen other than the use of language.

The entailment concerning the presence of a director is an interesting one, in that it is never explicitly indicated in the interface. Such a concept does, however, give a specific role to the user: they act as the director of their own gallery on their computer. This does make sense, as the user is able to maintain and alter the collection of templates available in the Project Gallery.

An important entailment that is *not* present is that galleries almost always display *artworks*. It is clear that templates and wizards simply do not deserve this kind of status in the interface. An important question, however, is whether users will be able to ignore the "artwork" entailment, given its central importance to the concept of a gallery.

Finally, examining these entailments raises the question of whether a gallery metaphor is truly present at all. Other than the use of language to indicate it, there is no specific suggestion of the metaphor in the interface. In particular, the entailments above could easily suggest other kinds of collections such as those found in a mail-order shopping catalog, for example. The key point here is that the actual *gallery* concept does not appear strongly enough indicated in the representamen. That said, the use of language does concretely state it, despite the lack of reinforcement via the UI metaphorical entailments.

User's Interpretant The only way to truly assess the user's interpretant is through user testing of some kind. The following analysis forms a kind of brief cognitive walk-through [57], and shows how some expectations have not been met. Overall, it seems that a user who took the gallery metaphor seriously would quite possibly be disappointed with its actual implementation in the interface. As mentioned, there is very little in the interface to suggest a gallery metaphor, other than its explicit statement in language and some functional details.

The metaphorical entailment that the contents of the gallery are *art*, although certainly not intended, could prove problematic for new users who take it seriously.

Finally, there *are* other kinds of galleries such as "shooting galleries" and galleries of *people* which could possibly be part of the user's interpretant. The likelihood of this is reduced, however, by the presence of the catalog metaphor, which fairly precisely indicates the kind of gallery intended.

Comments As suggested in the UI metaphorical entailments discussion, it is questionable whether this metaphor was really needed in the first place. It does appear as though any metaphor for a collection of artifacts would have served the purpose here, especially given that many such collections *do* have catalogs, such as shopping-malls. Additionally, there is no obvious reason as to why the gallery metaphor could not have been omitted, and only the catalog metaphor used. Both have many of the same entailments, but the catalog metaphor does not necessarily raise the problematic issue of "art."

6.4.3 Catalog

Metaphor As already suggested in the first case study, the metaphor here seems to be:

THE VIEW OF THE COLLECTION OF PROJECTS IS A CATALOG

The metaphor uses the concept of a catalog in order to explain the visual formatting of the view of available templates and wizards. It is entailed by the gallery metaphor and so enhances the coherence of the overall interface.

Metaphorical Entailments Some metaphorical entailments for the catalog metaphor have already been suggested in the prior case study, and these are repeated here.

- the view is made of paper.
- the view is a book.
- the view contains images of the projects along with labels.
- the view contains information on the projects.
- the view contains an introductory essay about the projects in the catalog.
- the view usually describes just one show.
- the view can contain prices for the items listed.

The metaphorical entailments of the catalog metaphor centre, as with the gallery metaphor, on its function or purpose. The aim of a catalog is to communicate the contents of a show or shows in a gallery to those who are interested in it. It is a source of *information* about those things in the gallery, though the kind of information offered can vary from catalog to catalog.

There are also entailments as to the physical nature of a catalog, which essentially pertain to its being a book with pages. These entailments are quite specific, but are an important part of what a catalog is in the real world.

Note that the various entailments about shows and so forth can be assumed because the presence of the gallery metaphor determines what *kind* of catalog is intended. That is, the gallery metaphor changes the possible interpretation of the catalog metaphor to being that of a *gallery* catalog, rather than a clothing or hardware catalog, for example.

Designer's Interpretant The designer's interpretant here involves considering how using properties and features of an art catalog might aid in understanding a collection of templates. The most obvious link to make is that the templates can be thought of as the individual artworks in a show. This in turn leads to considerations of representing the templates and wizards in a similar fashion to artworks in a catalog. Further thought leads to the idea of using images for displaying the templates and wizards. Additionally, thoughts as to the particular layout of catalogs might have lead to the layout which involves images along with captions. Perhaps, even further, the idea of having more detailed information as in catalogs was explored. Catalogs often display technical information about artworks such as their dimensions and composition, and perhaps something similar could be used in the user-interface. Templates might have their file-size or number of pages listed, while wizards might have the number of steps required to complete the task. Essentially, one can imagine a brainstorm on how the catalog concept could be used to represent a collection of templates and wizards in an accessible manner.

Representamen The most obvious part of the representamen for the catalog metaphor is language. In the "View" drop-down menu, one of the options is explicitly named "Catalog."

Perhaps more important, though, is the graphical aspect of the representamen. In particular, the visual layout of the "View" pane of the Project Gallery window is based upon the layout of a catalog. Each template is indicated via an image or icon of the template, and a caption which gives the template's name.

In a sense, the *function* of the catalog view also represents the catalog metaphor. The view displays the collection of templates and wizards in the Project Gallery for users to examine and choose between, which mirrors the functionality of a real world catalog. Therefore, the purpose of the catalog view can be said to be a part of the representamen, in some sense.

UI Metaphorical Entailments As already discussed in the first case study, the number of UI metaphorical entailments is quite reduced from the metaphorical entailments:

- the view contains images of the projects along with labels.
- the purpose of the view is to gain an overview of the projects available in the gallery.

As these entailments show, much of the detail involved in the concept of a catalog has been removed, leaving only the basic function and graphical layout. This is fairly standard practice in user-interface metaphor, as too much visual detail derived from the metaphor can be distracting. Note, for example, that the implementation of "pages" and the ability to "turn" them in the catalog metaphor might well have been more confusing than useful.

Importantly, the entailments above *do* seem to be enough to define the content of a very simple catalog metaphor. This is chiefly because the function of a catalog is what is most important about the concept. Additionally, the notion of a *gallery* catalog has been maintained by using the visual layout common to catalogs of artworks. Note, however, the problem already suggested in section 6.4.2 that the gallery and catalog metaphors in combination suggest that the contents are *artworks*, which is inappropriate to the interface.

It is interesting to note that entailments suggesting that catalogs contain more detailed information about the contained items could have been implemented, and might even have been useful. For example, as mentioned above, perhaps the number of pages of a template might have been indicated, or even a more informative description of what a template or wizard is used to accomplish, beyond simply its title. Additionally, it *could* be claimed that the metaphorical entailment concerning catalogs having introductory essays *has* been used in some sense: if the user selects a category which has no immediate templates to display,

then some information about the Project Gallery is displayed to them, along with some instructions for use.

User's Interpretant From the above discussion it can be predicted that the catalog metaphor should be relatively successful. In fact, as mentioned in the earlier section on the gallery metaphor, it seems as though the catalog metaphor could quite easily replace the gallery metaphor.

An important observation is that, because the catalog metaphor is quite minimally represented, expectations as to detailed entailments should not be too high. At the same time, *enough* of the concept has been implemented that the catalog metaphor does seem present, especially in the case of the graphical layout used.

The Model User would presumably note the use of language for "Catalog" and, from this, realise that the representation in the view pane resembles the format of a catalog. Since the catalog metaphor does serve the same sort of purpose as a real catalog, it should be satisfactory and thus the Model User will feel comfortable in dealing with it. Real users may, as has been noted several time, become confused by the reasonable assumptions that the templates and wizards are *artworks*, and that they ought to have *prices*, for example.

Comments Overall, the catalog metaphor seems a useful one in that it provides a sensible graphical layout for viewing the available templates and wizards. Entailments concerning the nature of the contents of the catalog and whether it is art may cause problems.

6.4.4 Wizard

Metaphor As established in the first case study, the metaphor here is that:

THE DIALOG BOX IS A WIZARD

This is an interesting metaphor in that the tenor (concept to be explained), "dialog box," is also metaphorical. This, then, shows the process of metaphorically explaining a user-interface metaphor, which allows an alternate or more detailed explanation to be given. A wizard is a special *kind* of dialog box, with a special purpose. Thus, while a dialog box is a place to have an exchange of information with the interface, a wizard involves an exchange of information with a specific kind of entity. This extra detail implies various things, most importantly that social conventions should apply.

Entity metaphors have traditionally presented problems in user-interface design concerning social conventions and general human psychology. Byron Reeves and Clifford Nass discuss this aspect of interfaces in their book *The Media Equation* [70]. In it, they perform various psychological experiments which indicate that humans do treat computers like people generally, and that this is reinforced whenever the computer seems to act in a human way. The use of a metaphor which directly suggests some aspect of the interface is a person brings this sort of issue to the fore.

Metaphorical Entailments Some metaphorical entailments of this metaphor are listed below. Because a wizard is such a complex concept, there are very many entailments, and so no attempt has been made at completeness.

- the dialog box is an object.
- the dialog box is an entity.
- the dialog box is a person.

- the dialog box performs magic.
- the dialog box can be helpful or a hindrance.
- the dialog box has a beard.
- the dialog box has a wand.
- the dialog box is very powerful and is not to be trifled with.
- the dialog box is extremely knowledgeable.

The first thing to note here is that the wizard metaphor builds upon several ontological metaphors, namely *object*, *entity*, and *person*. All of these metaphors have a great number of basic entailments which the wizard metaphor therefore takes on. This is especially true of the person metaphor, which has many entailments concerning social interaction and general psychological complexity.

The extension to wizard also clearly adds some fairly undesirable entailments to the metaphor. The use of the wizard concept adds entailments concerning their sometimes cryptic nature, and their fickleness, for example. Additionally, wizards can generally be either good or evil, or something in between, yet anything other than a good wizard would be a serious mistake in a user-interface.

Cultural considerations are important when examining the wizard metaphor. Different cultures certainly have different basic ideas about what wizard is and does, most importantly concerning the *kind* of person they are. Although Western culture largely views wizards as helpful and friendly, this is not guaranteed. Note, for example, that in some versions of the bible, wizards are, in fact, reviled:

> A man or a woman who is a medium or a wizard shall surely be put to death.
> They shall be stoned with stones; their blood shall be upon them.
> Leviticus 20:26-28

Nonetheless, the wizard metaphor does contain several entailments that would seem useful for explaining the purpose of the dialog box, which is to assist the user in achieving a complex task. These are particularly those entailments that deal with the wizard's intelligence and ability to achieve difficult objectives by performing magic. The key to this is that the person whom the wizard performs magic for need not understand the process at all, but only accept the result.

Designer's Interpretant The designer's interpretant likely involves mental images of what wizards look like and what they do. Additionally, research has hopefully been performed into cultural interpretations of wizards for any cultures the interface might be presented to. Consideration should also be given to the ways in which a dialog box might be made to represent a wizard in some way. This is a fairly interesting task, which would require considerable creative thinking. Possible ideas include an actual wizard figure which talks to the user, or a more simple layout which asks the users questions in text and permits them to enter replies. The latter is a technique already used in today's user-interface wizards, while the former should become possible in the future. "Clippy" from Microsoft Office is an example of a more personified assistant in the user-interface which takes the entity metaphor a step further than usual.

Representamen The wizard metaphor is represented most obviously through language. Certain options in the catalog view have labels explicitly including the word, such as the "Envelope *Wizard*" and the "List *Wizard*." They serve the additional purpose of indicating what the wizard's "speciality" is. Conceivably another entailment of wizards is that they have particular things they are good at.

The wizard metaphor is also represented graphically in the interface. This is through the image of a wand performing magic over a document or sheet of paper. The wand is interesting in that it is a form of *metonymy*. The wand is an item commonly associated with wizards, and so is part of a TOOL FOR THE USER OF THE TOOL metonymy. It is also important to see that the magic being emitted from the wand, and the wand itself, is light in colour, presumably representing "good" magic.

Finally, the dialog box itself, which is the wizard, or at least a means of communicating with one, must be considered as a part of the representamen for the metaphor. In the case of the envelope wizard, the dialog box appears to have little or no relation to the choice of metaphor. The dialog box simply presents text fields for the user to enter in sender and recipient addresses, and some options for the final formatting of the envelope. The list wizard, on the other hand, does contain some effort to implement the wizard metaphor. In particular, the first screen of its dialogs gathers information by using questions in a pseudo-conversation. This helps to enforce the person metaphor somewhat.

UI Metaphorical Entailments The UI metaphorical entailments for this metaphor are a considerably smaller set that the original metaphorical entailments. Examining the representamen leads to considering these as the chosen UI metaphorical entailments:

- the dialog box performs magic.
- the dialog box has a wand.
- the dialog box is extremely knowledgeable.
- the dialog box is good.

Note that the large part of the entailments concerning the wizard's status as a person have been eliminated. This is especially true of the social considerations that one might expect. The metaphorical entailment that a wizard might be good or bad has been defined more closely in the UI metaphorical entailments by the presence of the wand casting "good" magic.

User's Interpretant A key to the user's interpretant in this case seems to be establishing whether the elimination of the "social interaction" metaphorical entailments has any negative effect on users. This can only ascertained through specific user testing. Testing for issues such as this could be based upon the work of Reeves and Nass in *The Media Equation* as mentioned earlier [70]. They performed experiments which examined social concepts such as interpersonal distance and the role of experts as they applied to interactions with computers. The general finding was that results were similar to person-person interaction, indicating that social convention has an important role to play in user-interfaces.

It seems that the general design of the wizard metaphor indicates that the metaphor should not be taken overly seriously. In particular, there are very little of the *interactive* properties that would generally be expected of a wizard. Perhaps the only actually implemented interactive property is that the wizard has the power to accomplish complex tasks for the user. This is a fairly abstract entailment, and is hardly unique to a wizard. Thus, it

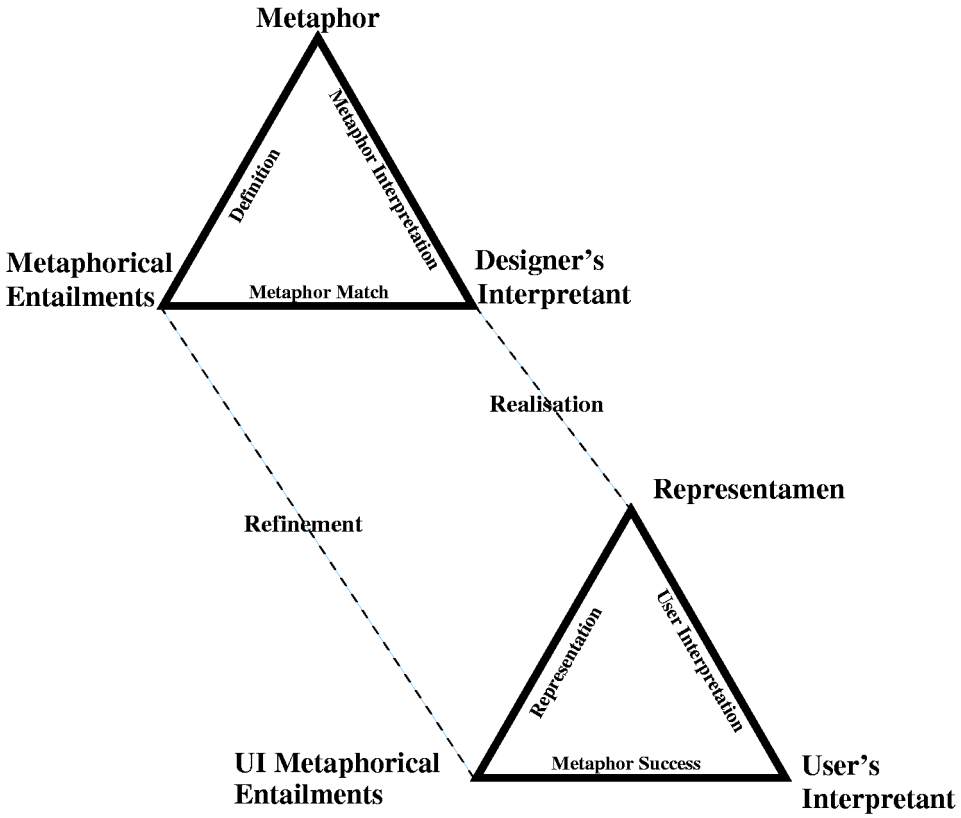


Figure 6.2: The semiotic model of user-interface metaphor as developed in chapter 4.

is questionable as to whether the metaphor has been successfully applied, and this may cause user confusion.

6.4.5 Results

The following sections will discuss overall insights gained from the examination of the gallery, catalog and wizard metaphors using the semiotic model. Before this discussion, the diagram of the semiotic model is reproduced in figure 6.2 to assist in remembering the general idea.

The basic working of the model is as follows. The *metaphor* is defined by a set of *metaphorical entailments*. These metaphorical entailments can be said to be the basic meaning of the metaphor. The *designer's interpretant* is the collection of thoughts that a user-interface designer has about the possibilities of the metaphor in question. This leads the designer to create the *representamen*, which is the representation of the metaphor in the actual interface. The representamen specifically represents the *UI metaphorical entailments* which are the refinement of the metaphorical entailments. The UI metaphorical entailments are the actual meaning of the metaphor as implemented in the system. Finally, the *user's interpretant* is the collection of thoughts had by a user who actually interacts with the user-interface metaphor via the representamen.

With this in mind, the results of the case study of the semiotic model will now be discussed.

Comparing sets of metaphorical entailments

One of the most interesting means of using the semiotic model for analysis is the analysis of the refinement relation as indicated in the figure. The comparison between the two sets of entailments allows considerable insight into the development of the user-interface metaphor

because it indicates the overall design process in a simple fashion. By understanding the process of eliminating the metaphorical entailments not present in the UI metaphorical entailments, it is possible to reflect on some of the major design decisions made.

Note, however, that this is not a perfect process in this case, because it is being applied retrospectively. This means that the UI metaphorical entailments are established by examining the representamen, and the metaphorical entailments are established by simply listing as many as possible. Clearly, both of these processes are in no way a precise analysis of the actual designers' thoughts. Nonetheless, it has been shown that certain insights *can* be gained using this approach. Note, for example, the considerable discussion of the omission of the "social interaction" entailments from the wizard metaphor, or the issues surrounding the "artwork" entailment of the gallery and catalog metaphors.

Comparing UI metaphorical entailments and representamen

By examining the link between UI metaphorical entailments and the representamen (the representation relation) it becomes possible to comment on how well the representamen is designed. In particular, analysis can be given as to how well it reflects the functionality suggested by the metaphor, and also how well it discounts functionality not intended by the metaphor. Note here, however, that because, in this retrospective analysis of the interface, the UI metaphorical entailments are *derived* from the representamen, it is more difficult to comment on their relationship. Nonetheless, interesting insight was gained into the *way* in which the representamen indicated certain entailments, such as the casting of "good" magic by the magician's wand, and the use of the graphical layout of a catalog.

User's interpretant supporting further testing

Theorising as to how the user's interpretant *might* be formed can give valuable guidance as to how user testing might be performed. Specifically, the hypothetical user's interpretant may well suggest particular directions for testing. This is especially so because the user's interpretant will be based on previous analysis of the representamen and UI metaphorical entailments, as discussed above. By uncovering possible issues here, the user's interpretant can be used to guide actual empirical testing of problem areas.

The process described above compares favourably with Nielsen and Mack's work on Cognitive Walk-throughs [57]. The similarity is that the process in this retrospective analysis involves examining the representamen from a critical perspective to uncover potential issues. The semiotic model is more detailed in the sense that it provides the specific concept of metaphorical entailments in order to make concerns more detailed.

Metaphors as representamens

A final discovery, in the analysis of the gallery metaphor, was that metaphors can be a part of other metaphors' representamens. In particular, the catalog metaphor features in the representamen of the gallery metaphor. This is an important means of analysing the coherence of metaphors. If one serves actively to help represent another, then their coherence is significant. This occurs when one metaphor entails another, which is then implemented also.

6.5 Conclusion

Over the course of this chapter numerous metaphors from the Project Gallery interface in Microsoft Office have been examined in depth. A fairly large number of metaphor were

identified and classified according to the material in chapter 3 on Lakoff and Johnson. In section 6.3 a large number of general insights into the workings of the metaphors in the chosen user-interface were expounded. The general conclusion concerning the first case-study is that the material from chapter 3 is extremely useful in providing a significant amount of interesting analysis of metaphors.

The semiotic model from chapter 4, although used outside its intended application to the design process, did reveal a number of important points about the user-interface metaphors in the Project Gallery. Besides this, it provided a solid framework for the detailed discussion of user-interface metaphors, and was shown useful for this task. It is expected that the application of the semiotic model *during* design would be very valuable, given the encouraging results found here.

The essential point demonstrated in this chapter is that the material discussed in the thesis *does* provide a structured means for approaching the analysis of user-interface metaphors. In particular, the insight gained from Lakoff and Johnson and Peircean semiotics is significant. Detailed analysis of user-interface metaphors was proved possible, and made far more accessible via the models.

Chapter 7

Usability Heuristics

7.1 Introduction

The research in the previous chapters has uncovered a considerable amount of detail concerning the structure and main uses of user-interface metaphors. Additionally, the case study of chapter 6 helped to analyse the models from a practical perspective and showed some of the evaluative capabilities. Quite often, the information discovered has suggested particular ways of doing things when it comes to the use of user-interface metaphor. In particular, it is possible to derive some specific heuristics that can be used both as a guide during the development of a user-interface, as well as a tool for the evaluation of finished user-interfaces. This follows the example of Nielsen and his usability heuristics which are intended for use both for assessment and during the design lifecycle [57]. This chapter presents a set of such heuristics, along with discussion of their meaning and specific examples of their application. The complete set of heuristics provides a practical interpretation of the material discussed in this thesis, and should assist in a more confident approach to user-interface metaphors.

Each heuristic will presented in its own section with a short name, the full heuristic, discussion of the heuristic and, finally, some examples to show how it can be applied. In the course of this chapter user-interface metaphors will simply be referred to as "metaphors." Terminology from previous chapters will be used freely. Before the heuristics are listed, some discussion of the structuring of the chapter is provided.

7.2 The Heuristics

7.2.1 Structure

The heuristics in this chapter cover a fairly wide variety of considerations within the topic of user-interface metaphor, all derived from the work in the previous chapters. Although each heuristic effectively stands on its own, it is possible to roughly group the heuristics into their particular kinds. The following listing of the heuristics outlines the particular order in which the heuristics have been provided.

First of all, heuristics that deal with particular categories of metaphor are listed. In particular, there is a heuristic for both orientational metaphors, and for structural metaphors. Although ontological metaphors are not addressed here, there is some information on them in heuristic 8. Finally, a heuristic on metonymy is provided for completeness and because of its similarities with metaphor.

1 *When choosing metaphors based on real world objects consider the real world usability issues for insight into interface usability.*

2 *When using orientational metaphors, make sure they fit into the overall framework defined by that orientation.*

3 *Use metonymy to aid in the representation of metaphoric actions, being careful to choose an appropriate vehicle.*

Next are metaphors which specifically concern the novelty of metaphors. In particular, guidance is given on the use of new and conventional metaphors, as well as the recognition of metaphors that may be implicit in thinking about the problem the software addresses.

4 *When using new metaphors, carefully indicate the UI metaphorical entailments to the user.*

5 *When using conventional metaphors, be careful to indicate any deviation from the accepted set of metaphorical entailments.*

6 *Examine all available information on the software project to identify metaphors that are implicit.*

The following three heuristics directly deal with the control of metaphorical entailments. Guidance concerning the use of ontological metaphors is also placed in this section as it explicitly informs entailment discovery and choice.

7 *Aim to use as many metaphorical entailments as possible while still keeping a firm focus on actual functionality.*

8 *Use the fact that structural metaphors tend to extend ontological metaphors to aid in establishing available metaphorical entailments.*

9 *As far as possible, control the user's interpretation of a metaphor by being clear on the exact metaphor being used.*

The next heuristics specifically concern the representamen in the user interface and its relation to usability.

10 *Represent only those affordances of the vehicle which have an implemented function in the user-interface.*

11 *Make sure all media used to convey a metaphor agree with one another.*

Next are three heuristics which concern the ways in which user-interface metaphors interact with each other and with the system around them. This includes conflicts between metaphors, as well as their ability to hide functionality and the issue of functionality coverage.

12 *As far as possible make sure that metaphors are coherent with one another.*

13 *Ensure metaphors do not inhibit access to any functionality.*

14 *Always address the aspects of the system* not *explained by metaphors.*

The final four heuristics deal particularly with the place of the user in user-interface metaphors, and thus could be said to concern the user's interpretant. The first three heuristics here concern the type of person the user is, from their experience and culture to their proficiency with computers. The final heuristic offers a means to influence the interpretant through explicit documentation of user-interface metaphors.

15 *Harness the user's experience to help identifying useful vehicles.*

16 *Design metaphors appropriate to the cultures the interface is intended to be presented to.*

17 *Allow expert users to circumvent the metaphorical means of achieving tasks where such circumvention would increase task efficiency.*

18 *Carefully document all structural metaphors used in the interface.*

It should be fairly clear that the set of heuristics just presented provide comprehensive coverage of the material discussed in this thesis. In particular, each of the classifications of metaphor provided by Lakoff and Johnson have been specifically dealt with in the heuristics. Heuristics 1, 2, and 8 deal with structural, orientational and ontological metaphors respectively. Heuristic 18 also directly concerns the use of structural metaphors. Heuristics 4 and 5 cover new and conventional metaphor types, while heuristic 3 concerns the use of metonymy. Advice on the more general interaction of metaphors is provided in heuristics 12, 13 and 14.

The semiotic model has also led to particular heuristics, with metaphorical entailments and UI metaphorical entailments featuring heavily in the advice (heuristics 7, 8, and 9). Matters relating specifically to metaphor selection are dealt with in heuristics 15, 16, and 6. The user's interpretant is addressed implicitly in heuristics 15, 16, and 17 also, by way of discussion of kinds of users. Finally, the representamen is specifically addressed in heuristics 10 and 11.

1 Structural Metaphor Usability

Heuristic *When choosing metaphors based on real world objects consider the real world usability issues for insight into interface usability.*

Origin This heuristic comes from the discussion of structural metaphors in section 3.4 as well as the general concern with object usability in such research as Donald Norman's *The Design of Everyday Things* [63].

Discussion Metaphors in the user-interface often have some real-world object serving as the vehicle. Given that usability is a watchword in user-interface design, and that it often considered equally important for real world artifacts, it makes sense to consider the usability of the real world object that forms the vehicle of a metaphor. As Donald Norman has frequently pointed out, real-world objects often have considerable usability defects which can render them almost unusable. Transferring these problems to the user-interface in the form of a metaphor is highly undesirable, and so usability issues of the vehicle must be attended to. Where possible it is sensible to choose a highly usable real world object, but it is also perfectly acceptable to specifically *improve* on the object's usability in its metaphorical form. A final consideration is whether the usability of an object is actually maintained in the transition to the user-interface. Although it is almost certain that there will be differences, it also seems reasonable to expect that major usability wins and losses will, in fact, be mirrored in the interface. This suggests that accounting for real world usability is a worthwhile activity.

Examples An obvious example of this heuristic is that one should not use something already difficult to use in the real world, such as a watch with two buttons mapped to fifteen functions, as a vehicle. There is no reason to expect the watch will be any easier to use in the user-interface than it was in its physical form.

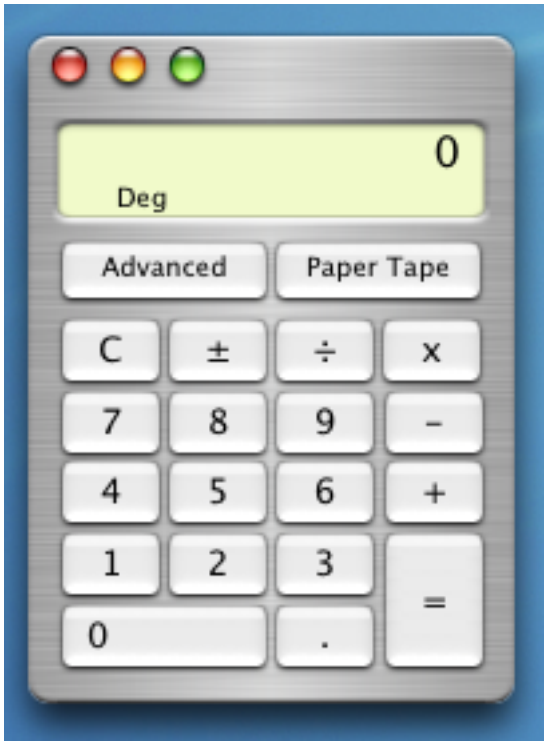


Figure 7.1: The standard calculator application from MacOS X.

Conversely, a successful application of the heuristic might be the implementation of a simple calculator to help the user perform standard mathematical operations, as shown in figure 7.1. The basic design of a calculator is a very usable one, and it can be expected that this usability will be transferred when it is used as a vehicle. It is, of course, possible to make the mistake of too literally transferring a complex calculator to the user-interface, as discussed by Mullet and Sano [52]. Such calculators are barely usable except by experts in the real world, and thus will be equally, if not more, complex in a user-interface. Note that it is very important to provide support for numeric keypad entry as well as mouse entry on the calculator to bring the interface closer to that of a real calculator.

2 Orientational Metaphors

Heuristic *When using orientational metaphors, make sure they fit into the overall framework defined by that orientation.*

Origin This heuristic is derived from the discussion of orientational metaphors in section 3.2.

Discussion The key to orientational metaphors can be found in the way that they link concepts together through a common orientation. Thus, all concepts utilising the vehicle "up," for example, are connected with each other. This connection, although simple, does influence the ways in which people think about the concepts. Associating something with a downwards direction automatically associates it with various negative ideas such as sadness and sickness in a Western culture.

Because, to some extent, the purpose of the user-interface is to control how the user perceives the system, any means of controlling association is important. In particular, it is

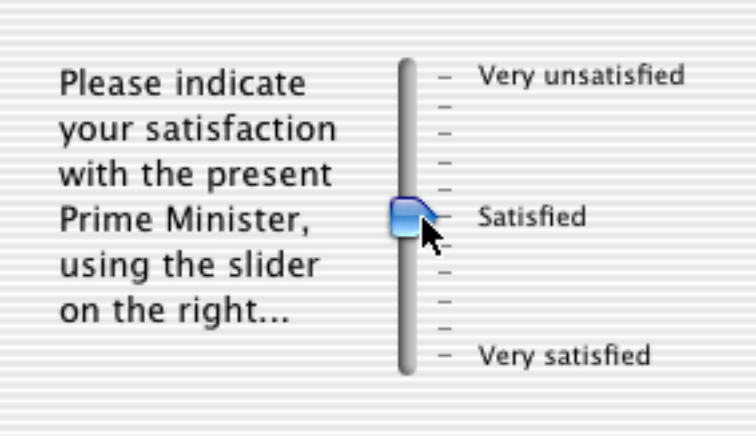


Figure 7.2: An example of misuse of orientational metaphor in a slider.

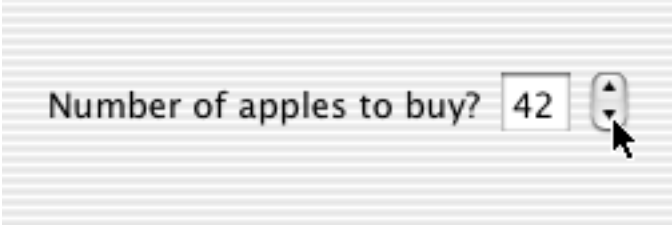


Figure 7.3: An example of using directional buttons to control a quantity.

crucial that designers be aware of any associations caused by the orientational metaphors in the system. Appropriate vehicles must be selected so as not to lead the user into thinking the wrong sorts of things about the interface. As already mentioned, it is difficult to incorrectly use orientational metaphors as the associations are so fundamental to most people.

Examples Consider an interface to a satisfaction survey concerning the Prime Minister, as in figure 7.2. If the sliders must be moved down to indicate greater satisfaction then significant confusion will arise. That is because in Western cultures the downwards direction is associated largely with negative concepts such as unhappiness and death. The slider should be moved *up* to indicate greater satisfaction.

Similarly, where there are two buttons for controlling numbers, one pointing up and the other down, as in figure 7.3, it must be the case that the upward pointing button *increases* the number, while the downward pointing button reduces it. That is because "up" is associated with *more* and "down" is associated with *less*. Any other means of approaching this will create confusion.

3 Metonymy

Heuristic *Use metonymy to aid in the representation of metaphoric actions, being careful to choose an appropriate vehicle.*

Origin Metonymy is discussed in some detail in section 3.8. It is also addressed implicitly in previous work on the semiotics of user-interface icons [9]. In particular, the metonymies CAUSE FOR THE EFFECT and THE EFFECT FOR THE ACTION are examples of indexical signs.

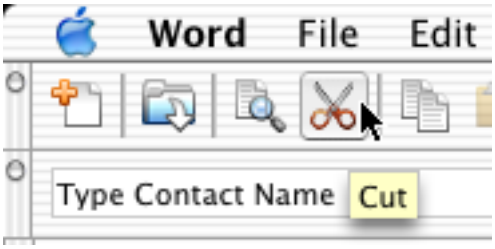


Figure 7.4: The metonymy used in Microsoft Word to represent the "cut" action.

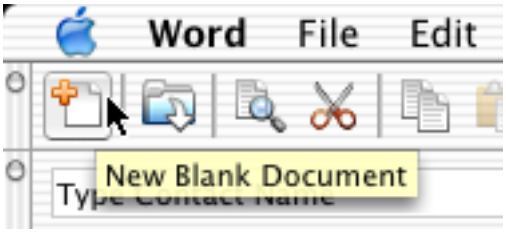


Figure 7.5: The metonymy used in Microsoft Word to represent the "create new document" action.

Discussion The key to metonymy in the user interface is its ability to indicate actions available using only a static representamen. The representation of an action is a difficult thing, because any action generally involves a process, rather than a single event occurring. Metonymy solves this problem by allowing either an effect, or a cause to stand for the whole action. Since the effect and cause can be displayed using a static image, this allows an action to be more easily indicated in a user-interface. The main issue, as identified in this heuristic is the selection of an appropriate vehicle for the metonymy. That is, selecting the thing which will represent the action in the interface. The choice here is crucial because it is all that will communicate to the user what the action is.

Examples In Microsoft Word, a small image of a pair of scissors is used to label a button for performing the "cut" operation on some text (see figure 7.4). Clearly, this is the metonymy THE SCISSORS FOR THE CUTTING which is an instance of THE CAUSE FOR THE EFFECT. This sort of metonymy is very common in the user interface, another example being the print button in Microsoft Word, which displays a small image of a printer, the cause of the act of printing. The new document icon in Microsoft Word displays a small piece of blank paper: the new document (figure 7.5). This is an instance of the EFFECT FOR THE ACTION metonymy in the form of THE NEW DOCUMENT FOR THE ACT OF CREATING A NEW DOCUMENT.

4 New Metaphors

Heuristic *When using new metaphors, carefully indicate the UI metaphorical entailments to the user.*

Origin This heuristic relates directly to the earlier discussion of Lakoff and Johnson's category of new metaphors in section 3.7. Additionally, it uses the concept of UI metaphorical entailments discussed in section 4.8.1.

Discussion The key feature of new metaphors is that users are unfamiliar with their set of UI metaphorical entailments. Note that new metaphors will almost always be structural,

Figure 7.6: The MacOS X "Traffic Light" window controls.

because ontological and orientational metaphors tend to be very basic and have vehicles that are reused in different metaphors. Because new metaphors are unfamiliar, the representamen must be even more carefully designed to indicate the UI metaphorical entailments which are intended in the interface. Along with this careful indication of the entailments, it may also help to have explicit documentation of the entailments to reinforce the message to the user (see heuristic 18). This includes indicators such as "tool-tips," which enable the designer to overtly state the purposes of some interface elements as in figures 7.4 and 7.5. Additionally, online help systems and tutorials can be used to explain, in a more direct fashion, what sorts of metaphorical entailments there are available. Finally, it is possible to have the documentation of the system, both online and off-line detail the ways in which a metaphor can be used in the system (for more on this, see heuristic 18). All of these techniques can serve to clarify just which metaphorical entailments are available to the user in the set of UI metaphorical entailments.

Examples In the MacOS X operating system the new metaphor of a traffic light is used to help explain the functions of the window controls for closing, minimising and maximising (figure 7.6). The metaphor is based on the idea that the user will understand the relation between the functions of the buttons and the functions of the different lights on a traffic light. Note that the designers additionally provided small symbols which appear on the buttons when the cursor is over them. These are intended to aid the user in establishing the functionality in a way alternative to the representamen of the metaphor. Note also, however, that the metaphor is apparently completely undocumented in the system help files.

5 Conventional Metaphors

Heuristic *When using conventional metaphors, be careful to indicate any deviation from the accepted set of metaphorical entailments.*

Origin This heuristic relates directly to the earlier discussion of Lakoff and Johnson's category of conventional metaphors in section 3.7.

Discussion Unlike new metaphors, conventional metaphors *do* have an established set of UI metaphorical entailments which are consistently used. Because of this, users have particular expectations when they encounter a representamen suggesting a metaphor conventional to them. What this means is that any deviation from the standardised set of metaphorical entailments must be well indicated to avoid confusion. Any additional entailments which are not part of the convention must be carefully conveyed to the user in the manner discussed for new metaphors and their entailments above. The more pressing issue, perhaps, is what to do when leaving *out* an entailment which the user expects as part of the conventional metaphor. In this case, it may be possible to indicate the absence of the entailment through a well-designed representamen, or to indicate in the documentation that the function is not present.

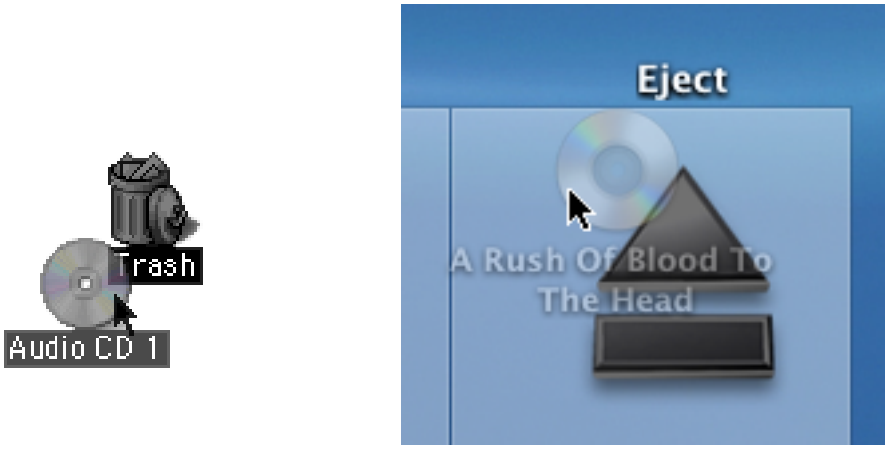


Figure 7.7: The action required to eject a CD-ROM in MacOS 9 and in MacOS X respectively.

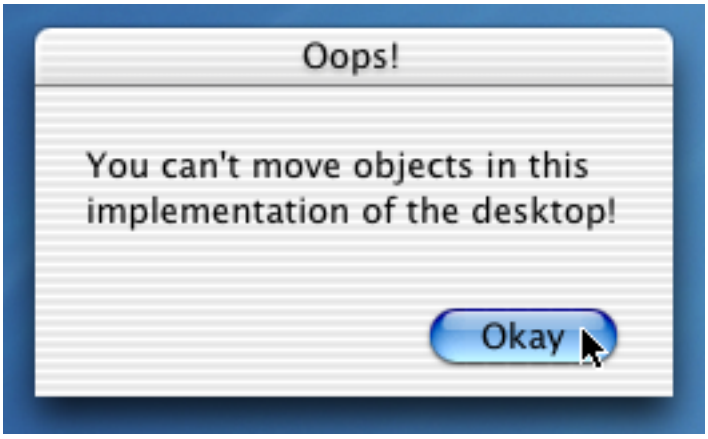


Figure 7.8: An example dialog which forbids the movement of objects on the desktop.

Examples Consider the quite strange entailment used in earlier versions of MacOS that putting a disk or CD-ROM in the trash ejects it from the computer (the first part of figure 7.7). This issue was amended in MacOS X where, once the icon of a disk or CD-ROM is being moved with the mouse, the trash-can icon changes into the symbol for ejection, thus indicating the function to the user (the second part of figure 7.7). This modifies the representamen in a way to clearly indicate the non-standard functionality to the user.

Conversely, as an example of removing a standard entailment, consider the idea of removing the ability to move files around on the desktop. One way to achieve this is to make it impossible to select a file's icon without bringing up a dialog which informs the user the function is unavailable (figure 7.8). In this way the representamen is altered to make sure the conventional behaviour is obviously not available. Naturally, removal of such a valuable feature is a terrible idea, but it does serve as an example nonetheless.

6 Implicit Metaphors

Heuristic *Examine all available information on the software project to identify metaphors that are implicit.*

```
A. Every entity has a six character long identifier.  The
identifier of the entity can be given as a command prompt parameter
or in a configuration file.

The location data (coordinates (x, y, z) range 0..100) will be
given from user interface.  User has to be able to change the
location data while the program is running.  After the location
has been changed, the entity will send new location data to other
members of the multicast group (as described below).  The location
data (x, y and z) are given as integers between 0 and 100.

Multicast address and port number is given by user or it is read
from the configuration file (default address = 239.255.0.0 and
default port = 4446).
```

Figure 7.9: Part of the specification for some network software for a computer science course assignment [65].

Origin This heuristic is based on the suggestion by Lakoff and Johnson that humans think in metaphorical terms. This immediately suggests that metaphors will be implicit in much of the early software process. A similar heuristic is suggested by Thomas Erickson [32].

Discussion There is strong evidence to suggest that people think in metaphorical terms quite commonly [43]. Given this, it is reasonable to expect that there will be metaphors implicit in any work done prior to the actual design of the user-interface. Implicit metaphors may appear in documentation such as the problem description, system requirements, and specifications, to name a few. Additionally, the approaches taken by developers in order to solve problems and present functionality may well contain implicit metaphors.

By examining all of these things, then, it ought to be possible not only to gain a rich supply of metaphors to consider in interface design, but also to obtain metaphors which are inherently relevant to the understanding of the software for those involved in its creation. The most obvious approach for identifying metaphors is to look for concrete nouns used in the documents and to consider whether they refer to real world things and objects. Similarly, examination of verbs used in the documents may uncover the use of real world actions that imply metaphors.

Examples As an example of implicit metaphors, consider the specification for a computer science assignment concerning network programming in figure 7.9.

In this small part of the specification of a program there are several implicit metaphors involved, illustrating the point of Lakoff and Johnson that people simply express themselves metaphorically. Some of the metaphors involved are:

Entity "Every entity has..." clearly provides an ontological entity metaphor.

Configuration file "... or in a configuration file..." indicates a "file" metaphor familiar to users, and of a particular sort.

Location "The location data..." This suggests that there may be orientational metaphors involved. This fits well with the existence of an entity, which normally has a location in space.

Program as Entity "... while the program is running." This indicates an entity metaphor applied to the program itself.

Group "... multicast group" This gives the metaphor of a group, which is generally a collection of *things* in the world.

Address "Multicast address..." This suggests a metaphor of an address, which is a concept traditionally applied to buildings in the real world.

Port "... port number..." This indicates a metaphor involving a port, which is traditionally a place of docking for ships in the real world.

There are likely more metaphors in the above specification, but those already outlined indicate just how many can be found with some ease. Note that this heuristic does not compel the designer to *use* all of these metaphors in the interface, but only to be aware of them as a potential source of design considerations.

7 Entailment Selection

Heuristic *Aim to use as many metaphorical entailments as possible while still keeping a firm focus on actual functionality.*

Origin This heuristic is derived from the discussion of metaphorical entailments in sections 3.6 and 4.8.1.

Discussion When selecting which of the metaphorical entailments will be transferred to the set of UI metaphorical entailments it is important to balance the desire to make the metaphor as close to the vehicle as possible, in order to increase user comfort, and the necessity of choosing entailments which reflect actual functionality. Thus, include as many of the metaphorical entailments as possible, while still not going overboard and including entailments which imply functionality not present in the system. Naturally, if, while thinking of metaphorical entailments, one suggests something that *does* seem a useful function, this function ought to be added. This process, therefore, additionally serves as a useful review of system functionality.

Note that purely perceptual aspects of the vehicle are safe to use in the representamen, so long as it is done so with reference to heuristic 10 which deals with the representation of affordances. Any aspects of the vehicle which deal with actual functionality should only be included in the representamen if the functionality exists in the system.

Examples As an example of not implementing entailments unreflected in functionality, consider that in real life one can knock over a trash-can, causing its contents to spill out. Although, this *could* be a UI metaphorical entailment, it is clearly a useless one and should be left out. Figure 7.10 shows what it might be like to include this entailment in the user-interface.

Conversely, the ability to take something back out of a trashcan in the real world *is* a useful property and so this should be reflected in the UI metaphorical entailments of the metaphor, as it is in today's interfaces. See figure 7.11 for an example of this functionality.

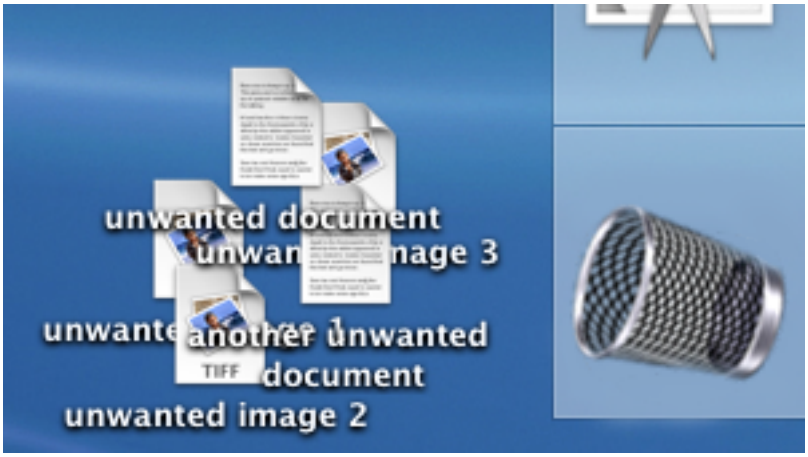


Figure 7.10: What it might look like if one knocked over the trashcan in MacOS X.

8 Metaphor Extension

Heuristic *Use the fact that structural metaphors tend to extend ontological metaphors to aid in establishing available metaphorical entailments.*

Origin This heuristic derives from the relationships between structural and ontological metaphors as discussed in section 3.5.and observed in the case study of chapter 6.

Discussion Structural metaphors are often extensions of lower-level ontological metaphors. Because of this, it is possible to divide the work of identifying metaphorical entailments to some degree. By acknowledging the underlying ontological metaphors, a designer can extract a large number of basic metaphorical entailments, independent of consideration of the overlying structural metaphor. Once this is done, the more specific metaphorical entailments brought by the real world vehicle of the structural metaphor can be analysed in isolation. This will make it easier to focus on finding as many as possible, which, in turn, should lead to more useful entailments being found.

By viewing the structural metaphor as being decomposable into ontological metaphors along with an extra structural part, the task of identifying metaphorical entailments becomes a more routine and structured activity. Additionally, with experience, the set of the metaphorical entailments for frequent ontological metaphors such as *object* and *container* could become standardised, and thus this part of the work would be removed almost entirely.

Examples A folder on the desktop is an example of metaphor extension (figure 7.12). At its most basic, this can be considered as based on the *object* metaphor. Thus, it possesses certain entailments concerning physical existence such as having a size, position and so forth. Additionally, however, this is extended into a *container* metaphor. This adds entailments concerning the ability to store or contain something, having a bounding surface, and so on. Finally, the actual structural metaphor of a "folder" adds in very specific entailments, such as what the object *looks like* and the kinds of things the container *contains*.

9 Metaphor Clarification

Heuristic *As far as possible, control the user's interpretation of a metaphor by being clear on the exact metaphor being used.*

Figure 7.11: The act of taking a file back out of the trashcan in MacOS X.

Figure 7.12: The standard folder from MacOS X.

Origin This heuristic comes from consideration of the representamen as the key element of a user's encounter with a user-interface metaphor. The concept of a representamen is discussed throughout chapter 4. Specific discussion of the representamen as part of the model of user-interface metaphor is found in section 4.8.1.

Discussion Any given metaphor has a great number of metaphorical entailments which can be conceived by users. Some of these entailments will be not be applicable in the actual interface and so, as far as possible, it is desirable to be as specific as possible about the implemented entailments. In particular, it is always better to clarify any vague aspects of the metaphorical entailments in a metaphor by making the representamen as clear as possible.

The designers should make their definition of the metaphor itself as specific as they can, to avoid later confusion in the design process. A specific metaphor will help to define the metaphorical entailments that will be applicable.

Examples Consider the wizard metaphor as used in many applications. The concept of a wizard by itself is quite vague as there can be many *kinds* of wizards. Most crucially, there can be both good and bad wizards. Clearly, in a user-interface, the designer wishes the user to be thinking of *good* wizards only. Therefore, this must be specifically accounted for in the metaphor and indicated in the representamen. The metaphor, therefore, is not simply THE DIALOG BOXES ARE A WIZARD but that THE DIALOG BOXES ARE A GOOD, HELPFUL

Figure 7.13: The icon for a wizard in Microsoft's Project Gallery interface.

WIZARD. In turn, the representamen can be modified to show, perhaps, a smiling old man with his palms showing. Maybe he could be dressed in white, or depicted casting white magic (figure 7.13). These sorts of aspects of the representamen help the user to understand that the wizard is specifically a *good* and *helpful* one, rather than possibly being evil. Although this example might seem somewhat ridiculous, it is nevertheless true that the notion of a "bad wizard" might distract some users, especially younger users who are actively interested in wizards and the like.[1] Additionally, wizards can be suggested to have *specialties* such as the "Envelope Wizard" in the Project Gallery discussed in chapter 6.

10 Affordances

Heuristic *Represent only those affordances of the vehicle which have an implemented function in the user-interface.*

Origin This section comes from the combination of considering the nature of structural metaphors as based on real world objects, discussed in section 3.4 along with the affordance theory originally developed by J. J. Gibson [33], and re-applied by Donald Norman [63]. Additionally, discussion of the representamen in section 4.8.1 has influenced the discussion here.

Discussion Affordances are the aspects of an object that permit interaction of some kind [33, 63]. Given that vehicles can be real world objects, an awareness of the affordances involved is important. First, it can be used to ensure the affordances appropriate to the user-interface are displayed in a way which will enable the user to understand and utilise them. Second, this awareness can be used to prevent the inclusion of inappropriate affordances which will lead the user to believe there is functionality available which is *not* included in the UI metaphorical entailments.

The basic point here is that the affordances of the real world thing chosen as a vehicle can often be translated into the user-interface. Only those affordances which reflect UI metaphorical entailments are desirable because any others will suggest functionality not actually present. Examination of the affordances should also help in enumerating metaphorical entailments relating directly to functionality provided by the vehicle.

[1] Consider, for example the huge popularity of the Harry Potter books and movies, where the good wizard Harry Potter is pitted against the evil wizard Voldemort.

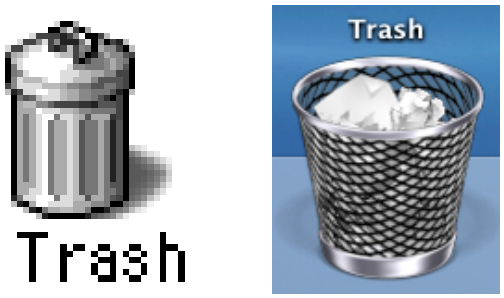


Figure 7.14: The trashcans from MacOS 9 and MacOS X respectively.

Example In MacOS 9 the trashcan had a lid on it (the first image in figure 7.14). This lid had a handle which affords "taking the lid off." This affordance, therefore, suggests a UI metaphorical entailment concerning the ability to remove the lid of the trashcan. This action, however, is not actually possible in the interface. This conflict between affordances and actually permitted behaviour could lead to confusion.

In MacOS X the trashcan no longer has a lid (the second image in figure 7.14), and so the affordance concerning lid-removal is no longer present. This accurately reflects the available UI metaphorical entailments.

11 Media Co-ordination

Heuristic *Make sure all media used to convey a metaphor agree with one another.*

Origin This discussion stems from the analysis of the representamen in section 4.8.1 and the point that the representamen can be made up of many types of media.

Discussion The representamen of a metaphor can be composed of various sorts of media, including graphics, animations, sounds, interaction, and more. Given that each of these aspects of the representamen can be important in conveying the overall metaphor it is crucial that they work *together* in a co-ordinated fashion. In particular, none of the aspects of the representamen should conflict with one another in any way as this is guaranteed to be disturbing to the user of the interface. A well co-ordinated representamen will reinforce the messages the interface is meant to convey and make the user's job of interpretation that much easier.

Examples As an exaggerated example of the lack of media co-ordination, imagine if on selecting "Empty Trash" in MacOS X the sound of a flushing toilet was played on the computer's speakers. This complete mismatch between the graphical and auditory aspects of the representamen would be confusing and distracting to users. As it is, no sound is used at all, though perhaps the sound of a dump-truck could be used if auditory feedback was desired.

In some interfaces, such as that for the GIMP, a graphical manipulation tool, there is the concept of a "tear-away menu." This is a menu which can be "torn" off the menu bar, and then placed somewhere convenient for the user. See figure 7.15 for an illustration of this interface concept. In the figure, the "File" menu has been torn off for convenient access and the "Dialogs" menu is about to be torn off also. This ability to tear the menu off might seem incongruous with the traditional view of the menu which is that it is attached to the menu bar. Such tear-away menus, however, tend to have perforations as part of the menu design,

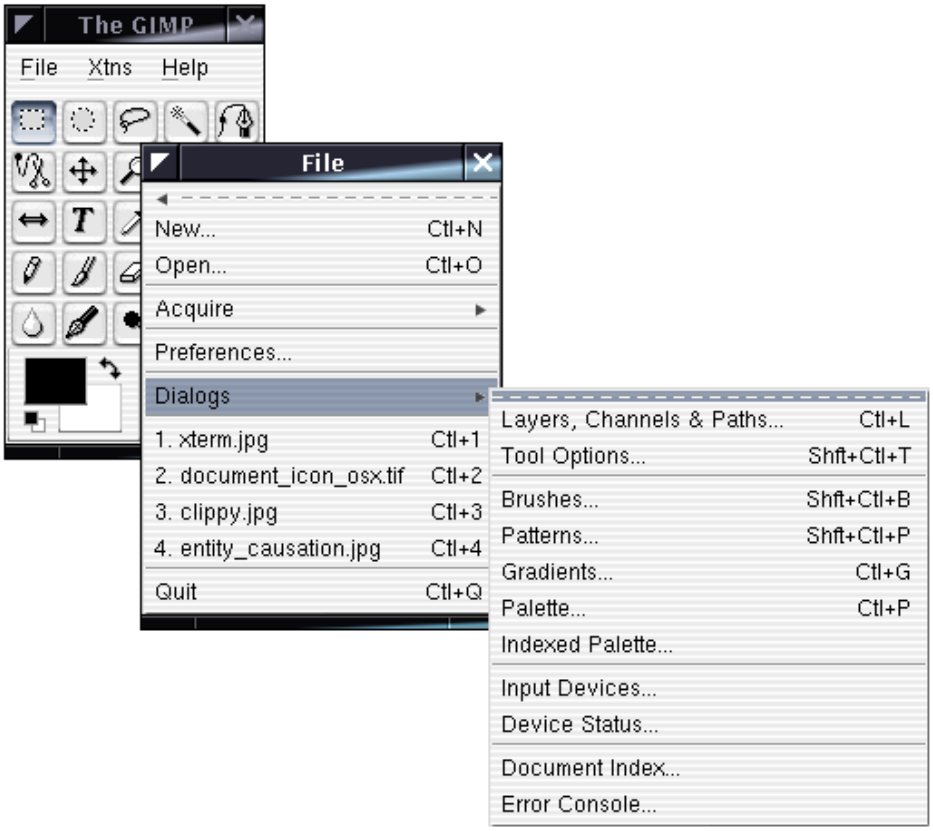


Figure 7.15: A "tear-away" menu from the Gimp in Linux.

which indicate that the action of tearing is purposefully allowed. Here the co-ordination of the visual and kinaesthetic senses is achieved fairly well.

12 Coherence

Heuristic *As far as possible make sure that metaphors are coherent with one another.*

Origin This heuristic arises from Lakoff and Johnson's discussion of the notion of metaphor coherence. Specifically, this involves the avoidance of areas of incoherence in the user-interface, brought about by conflicting metaphorical entailments as discussed in section 3.6.3.

Discussion When using a single metaphor, issues of coherence do not arise. Once multiple metaphors are employed in a single interface, however, there is the potential for problems between one or more of them. In particular, the metaphorical entailments of one metaphor can conflict with or contradict those of another. When this happens it will obviously lead to considerable confusion because the user will not know which of the metaphorical entailments they should expect to be actually applied in the system.

To check on coherence it is key to examine exactly what functionality various metaphors in the interface cover. Problems with coherence can only occur when two metaphors concern the same part of the interface. In that case, they may have entailments which do not work well together and cause incoherence. In this case, it is necessary to eliminate one of the metaphorical entailments in order to defuse the incoherence. The choice of which entailment to remove is difficult, but should be based on which of the entailments is seen as most valuable

in explaining and indicating the functionality in question. The removal of the entailment must be careful indicated in the representamen for that metaphor. Alternatively, an entire metaphor could be removed if it is too problematic.

Examples As an example consider the issues involved with the use of the metaphors "file" and "folder" in most common desktop user-interfaces. In the real world, a file and a folder are very similar things. In fact, a file is simply a folder with actual material (documents) in it. Thus, the notion of putting a file in a folder is a problem for coherence: the collected information of a file is already *in* a folder. A real file has the entailment "you can put documents in the file," and a folder also has this entailment. This incoherence has not actually been completely eliminated, but the use of the document metaphor has helped. Documents *can* be put in folders in the real world and thus the entailments cohere with those of a folder metaphor.

Similarly, the use of the "directory" metaphor in older interfaces is incoherent with their storage of files. A directory is most obviously a phone directory, which stores phone numbers. More generally, a real world directory is an alphabetical listing of names and other data. This entailment does not fit with the storage of "files" in "directories." The fact that people successfully work with files and directories shows that incoherence is not always a critical problem, but it should still be avoided where possible.

13 Metaphor Influence

Heuristic *Ensure metaphors do not inhibit access to any functionality.*

Origin This heuristic stems from the consideration of metaphorical entailments in sections 3.6 and 4.8.1.

Discussion Although metaphors are regarded as a good way to explain system functionality, it is also true that they can make certain functions seem inaccessible or inappropriate. Specifically, a particular metaphor can lead the user to believe certain things cannot be done in the system when, in fact, they can. Such situations must be noted and measures taken to avoid or remove the problem. Possibilities include re-implementing the function in a different way, or specifically documenting the way in which the function can be accessed.

Examples In the older versions of MacOS, a disk or CD-ROM was ejected by placing it in the trashcan on the desktop (first part of figure 7.16). Applying the trashcan metaphor, this would suggest that the user was throwing the disk or CD-ROM away and that they did not want it. This conflicts with the function of ejection, because in a such situation the user *does* want to keep the disk or CD-ROM, just not in the computer. Thus, the trashcan metaphor inhibits the action of ejecting from the computer. One option would be to have a menu option instead, and remove the function from the trashcan. Alternatively, in MacOS X the trashcan turns into an ejection symbol when a CD-ROM or disk is being moved (second part of figure 7.16). This makes it much clearer that the disk or CD-ROM will not be thrown away by the action, but simply ejected.

14 Non-Metaphorical Aspects

Heuristic *Always address the aspects of the system* not *explained by metaphors.*

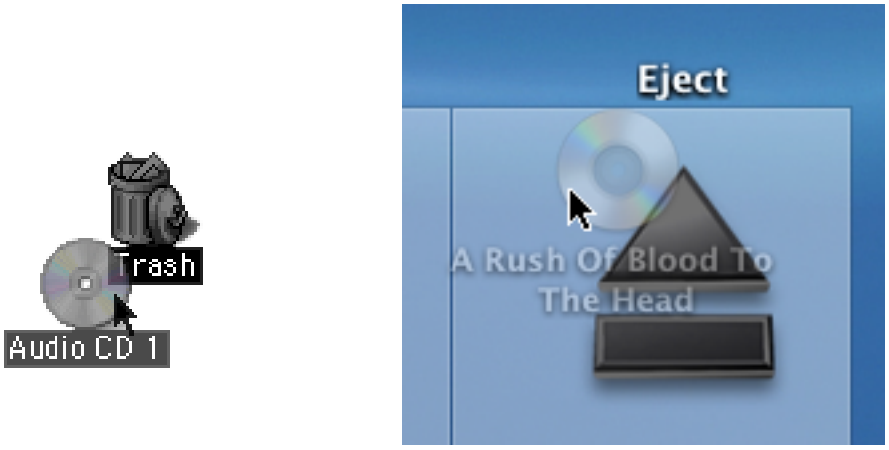


Figure 7.16: Examples of ejection from MacOS 9 and MacOS X respectively.

Origin This heuristic arises from the natural realisation that it is unlikely that every piece of functionality in a system can or should be explained metaphorically.

Discussion It must be acknowledged that there will be parts of the user-interface which are entirely non-metaphorical. In these cases, it is necessary to use traditional usability techniques to make the system as usable as possible. In addition, it should be attempted to make sure the metaphorical and non-metaphorical aspects of the interface work well with each other and do not conflict. This is important so as not to break the "metaphor illusion" the user is supposed to be working under.

Examples As an example, consider the document metaphor. While it does usefully account for a large number of details about the particular kind of file a document is, it still omits many things. The notion of explicit permissions is more or less absent from the traditional conception of a document, for example. Similarly, the ability to compress metaphorical documents is something that is absent in real world documents. These sorts of extra functionality must be carefully addressed so that they fit in with the metaphor coherently, while still being usable.

15 User Experience

Heuristic *Harness the user's experience to help identifying useful vehicles.*

Origin This heuristic is derived from the work of Jakob Nielsen who includes in his own set of heuristics the idea of making sure that the user is taken into account when developing software ("Know the User" [57]).

Discussion All users are people who live in the real world and thus have a wealth of real world experience to draw on when understanding things. Any experience of the real world can potentially be used as the vehicle for a metaphor, most especially experience with particular objects and entities. Therefore, it makes sense to closely examine the user's experiences in order to establish metaphors a user-interface.

In particular, it is advisable to pay considerable attention to the most common aspects of the user's life, as these should lead to vehicles which the user is most comfortable with. In order to obtain a good list of potential vehicles, it is crucial to consider exactly who the

target of the software is, so as to try to find the right sorts of vehicles. Possible ways of creating a good list are by interviewing potential users, brainstorming based on a general idea of the user, and possibly even using the concept of Personas developed by Alan Cooper [1, 20]. Under the Personas approach, the designers will create several imaginary, "typical" users and assess their common experiences for vehicles. Another possibility is to observe users as they work in their actual work environment, in order to more clearly establish what it is they do when working.

Given that the key advantage of user-interface metaphors is claimed to be the leveraging of existing user knowledge, it is clear that examining the users' world is a good place to start. The user's world is a rich source of potential vehicles for user-interface metaphors.

Examples As an example of using day to day experiences to brainstorm vehicles for a user group consider the use of air-conditioning. This could be a useful vehicle in some metaphor, but only for a particular set of users: those familiar with the use of air-conditioning units. The vehicle would be useless to users who have no experiences involving such technology. The key is to pick experiences of the actual users, rather than *any* experiences.

Considering a user's work environment could well lead to useful vehicles. In fact, this is demonstrated very well in the desktop metaphor which has obtained vehicles such as "desktop,' "files," "folder," and "trashcan." It should be clear that such vehicles could be easily obtained by observing target users at work.

16 Culture

Heuristic *Design metaphors appropriate to the cultures the interface is intended to be presented to.*

Origin This heuristic is derived from Lakoff and Johnson's considerations of the cultural influences on metaphor, as discussed throughout chapter 3.

Discussion This heuristic is essentially a specialised case of heuristic 15. An important area of any user's experience is the culture or group which they identify with. This culture or group will particularly have a large influence on the kinds of metaphors that user finds conventional or new. Because of this, it is especially important to acknowledge culture when attempting to develop metaphors for an interface.

Examples Taking the example from section 3.2 of the culture which uses a metaphor of the body to discuss orientation, it can be seen how culture might have an effect. Instead of referring to the top of the screen in the interface, for example, it would be more natural for that particular group from Mexico to refer to the "head" of the screen. This would then use a metaphor the group was familiar with. Acknowledging the metaphors of the target audience's culture in an interface can only be a good thing.

17 User Types

Heuristic *Allow expert users to circumvent the metaphorical means of achieving tasks where such circumvention would increase task efficiency.*

Figure 7.17: The menu which indicates the accelerator key-combination for moving files into the trash in MacOS X.

Origin This heuristic arises from the consideration of the existence of different *types* of user. Specifically, the distinction between expert and novice users as made by Ben Schneiderman and others [73, pp.68–69].

Discussion The use of metaphors in user-interfaces is normally intended as a way to increase accessibility and usability for *novice* users. There is a risk, however, brought up by various critics that this usability for beginners is at the expense of expert and possibly intermediate users [15, 54, 56]. The problem is that, while the metaphor may well provide an easier way for novice users to understand the system, it may not facilitate the most efficient ways of completing tasks. This will ultimately frustrate expert users if they are forced to work through the metaphor at all times.

Fortunately, it is not difficult to imagine circumventing the metaphorical approach for those users who desire efficiency, without interrupting the use of the metaphor by novice users. One particular idea which is relevant here is Nielsen's concept of "accelerators" in the interface. As Nielsen notes, typical accelerators include:

> abbreviations, having function keys or command keys that package an entire command in a single key-press, double-clicking on an object to perform the most common operation on it, and having buttons available to access important functions directly from within those parts of the dialog where they may be most frequently needed.
> [57, p.139]

Clearly, these sorts of suggestions apply well to the notion of getting around metaphorical ways of achieving tasks, while not obscuring the metaphor in any way. Accelerators should always be included in interfaces which make heavy use of metaphors. Note, however, that it is not guaranteed that metaphors will inhibit the speed of expert users in their tasks. In particular, some metaphors may well speed up *any* user-type's work.

Examples As an example of one of Nielsen's accelerators, consider the task of moving a file to the trash in MacOS. Using the metaphors of objects such as files and the container that is the trashcan, the user can pick up (click and hold) the file and then move it to the trashcan (move the mouse), then drop it (release the mouse button). An accelerator, which eliminates any mouse movement allows an expert to simply select the file and then hit a key-combination (command + shift + delete) which automatically moves the file to the trash (see figure 7.17).

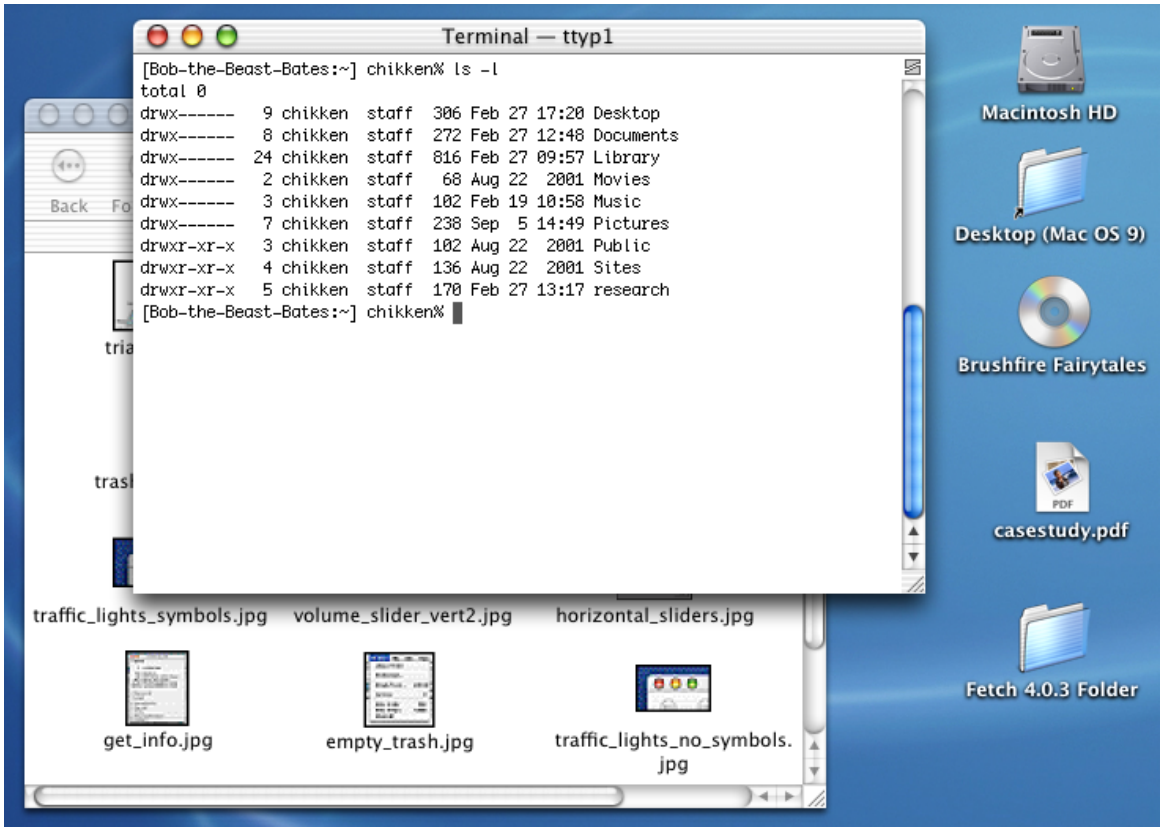


Figure 7.18: The command-line interface available on top of the standard desktop interface in MacOS X.

As another example of accounting for user levels, consider the availability of both the metaphorical desktop interface as well as a command-line interface in all of today's major operating systems such as Linux, Unix, Windows and MacOS X (figure 7.18). The command-line allows expert users to circumvent the mouse-oriented file manipulation of the desktop in favour of keyboard-based commands.

An example of a metaphor that is likely faster for experts than alternatives might be the template metaphor. This allows any kind of user to begin with a large amount of their formatting work already achieved. In particular, the power of creating personal templates for experts is likely to accelerate their work far greater than the ability to quickly, but manually, generate the same formatting every time they need it.

18 Documentation

Heuristic *Carefully document all structural metaphors used in the interface.*

Origin This heuristic is based on Jakob Nielsen's similar heuristic concerning software engineering in general [57]. Additionally, it relates to consideration of the user interpretation relation discussed in section 4.8.2 which suggests making things as explicit as possible.

Discussion Because metaphors can only be interpreted directly through their representamen, there is much room for misunderstanding. By documenting the exactly functionality offered by the user-interface metaphors in a system, this confusion can be addressed explicitly by the designers. Note that only structural metaphors really need to be documented:

ontological metaphors either tend to underly structural ones, or to be so subtle as to not really require specific documentation; and orientational metaphors are not overtly noticeable in a system and will also not tend to require documentation.

Examples As an example, the metaphor involved in the window controls of MacOS X should really be documented. The metaphor is that THE WINDOW CONTROLS ARE A TRAFFIC LIGHT and is complex enough that explicit explanation would likely prove useful to some users.

7.3 Conclusions

This chapter has presented a series of heuristics intended to aid in the development of user-interfaces which involve metaphors, as well as to provide the basis for evaluation of such interfaces. It represents the evaluative application of the material discussed in chapters 3 and 4 and thus demonstrates that the theory can be put into more practical terms. The examples given for each heuristic go some way to showing that they are useful and, in the following chapter, the heuristics are examined in detail in a heuristic evaluation of the Project Gallery.

Chapter 8

Heuristic Evaluation

8.1 Introduction

In chapter 7 a collection of heuristics intended to inform the design of user-interface metaphors was derived from the research in this thesis. The heuristics were based on the material in chapter 3 on Lakoff and Johnson's classification of metaphor, and chapter 4 on the semiotics of user-interface metaphors. Additionally, the case study performed in chapter 6 influenced the development of them.

Performing a heuristic evaluation is useful because it allows us to test the heuristics on a real user-interface. Although insight will certainly be gained about the metaphors in the user-interface examined, the key is to establish how easily and how successfully the heuristics can be applied in the real world. Although the heuristics from chapter 7 are largely intended to be used in the *design* process, it will be shown that a subset of them can be used to evaluate existing user-interfaces.

The structure of the chapter is as follows. First, the general setup of the evaluation in section 8.2 explains the heuristics chosen for the evaluation and the metaphors chosen to be evaluated, as well as discussing the process of the evaluation in some depth. Section 8.3 then reports the results of the evaluation. Section 8.4 provides analysis and insight gained from the study. After some thought on further studies and work is discussed in section 8.5, the chapter is concluded in section 8.6.

8.2 Evaluation Setup

8.2.1 Introduction

The Project Gallery was selected as the subject of the heuristic evaluation. This choice was made because a large degree of familiarity with the interface had already been gained via the study in chapter 6, making it easier to guide evaluators where necessary. Additionally, as already stated in chapter 6, the Project Gallery is of a manageable size, while still being varied and widely used.

8.2.2 The Heuristics

In chapter 7, eighteen heuristics were developed for the assessment of user-interface metaphors. These heuristics were based on the research into material by Lakoff and Johnson in chapter 3, and the detailed semiotic model of a user-interface metaphor in chapter 4.

The heuristics were largely intended for use during the process of interface *design*, and, because of this, not all of them were applicable to interface *evaluation*. In addition, it was

noted that the heuristics in chapter 7 use the technical language developed in the thesis. Although this vocabulary is useful because of its specificity, it was felt to be inaccessible to the evaluators who had not read the thesis. The option of explaining the terminology used was discarded as being overly complex. Instead, the heuristics were rewritten to be more accessible. The decisions made when selecting and rewriting the heuristics are discussed in detail below.

Selecting and Rewriting the Heuristics

The heuristics used in the evaluation process are listed in appendix A. This is the sheet given to evaluators during the testing of the Project Gallery. This section presents the heuristics from chapter 7 and briefly indicates whether or not they were included in the evaluation, and offers some commentary on the rewriting of the heuristics where relevant.

1. **Structural Metaphor Usability**

   Original Heuristic *When choosing metaphors based on real world objects consider the real world usability issues for insight into interface usability.*

   Evaluation Heuristic 1 *The metaphor should not transfer any usability problems from the vehicle (explaining concept) into the user-interface.*

   The heuristic was rephrased as a prescriptive statement to make evaluation more definite.

2. **Orientational Metaphors**

   Original Heuristic *When using orientational metaphors, make sure they fit into the overall framework defined by that orientation.*

   Evaluation Heuristic Omitted.

   This heuristic was omitted as overly complex for evaluators to come to grips with in a short space of time. Additionally, the Project Gallery does not contain many orientational metaphors, as indicated in chapter 6.

3. **Metonymy**

   Original Heuristic *Use metonymy to aid in the representation of metaphoric actions, being careful to choose an appropriate vehicle.*

   Evaluation Heuristic 2 *The metaphor should use relevant metonymy (one object standing in for a related object or concept) to represent possible actions in the user-interface.*

   The term "relevance" was used instead of "appropriateness," as it framed the issue more specifically for evaluators. The reference to "metaphoric actions" was removed to avoid confusion.

4. **New Metaphors**

   Original Heuristic *When using new metaphors, carefully indicate the UI metaphorical entailments to the user.*

   Evaluation Heuristic Omitted.

This heuristic was omitted because it is essentially a design-oriented metaphor. Although an evaluative interpretation is possible, the judgment of what constitutes a new metaphor is a difficult one for evaluators to make.

5. **Conventional Metaphors**

   Original Heuristic *When using conventional metaphors, be careful to indicate any deviation from the accepted set of metaphorical entailments.*

   Evaluation Heuristic 3 *Conventional metaphors should always indicate any deviation from the standard functionality or usage of the metaphor.*

   Minimal rewording was necessary.

6. **Implicit Metaphors**

   Original Heuristic *Examine all available information on the software project to identify metaphors that are implicit.*

   Evaluation Heuristic Omitted.

   This heuristic was omitted because it is strictly for the design process.

7. **Entailment Selection**

   Original Heuristic *Aim to use as many metaphorical entailments as possible while still keeping a firm focus on actual functionality.*

   Evaluation Heuristic 4 *The metaphor should use as many useful facts about the vehicle (explaining concept) as is possible without violating other heuristics.*

   This heuristic was reworded to replace "metaphorical entailments" with "useful facts." It was decided to phrase the conditional as "without violating other heuristics" rather than "keeping a firm focus on functionality." The latter was deemed too close to a designer's perspective.

8. **Metaphor Extension**

   Original Heuristic *Use the fact that structural metaphors tend to extend ontological metaphors to aid in establishing available metaphorical entailments.*

   Evaluation Heuristic Omitted.

   This heuristic was omitted because it was felt that ontological metaphors were too complex to explain in the time available. Additionally, the heuristic is intended more for designers seeking to establish metaphorical entailments, than it is for evaluation.

9. **Metaphor Clarification**

   Original Heuristic *As far as possible, control the user's interpretation of a metaphor by being clear on the exact metaphor being used.*

   Evaluation Heuristic 5 *The metaphor should be as precise as possible to avoid any confusion about its intended meaning.*

   The heuristic was reworded to simplify the language.

10. **Affordances**

    Original Heuristic *Represent only those affordances of the vehicle which have an implemented function in the user-interface.*

    Evaluation Heuristic 6 *The metaphor should not represent or suggest any functionality that is not accessible.*

    The heuristic was reworded to avoid discussion of the technical term "affordance."

11. **Media Co-ordination**

    Original Heuristic *Make sure all media used to convey a metaphor agree with one another.*

    Evaluation Heuristic 7 *The ways in which the metaphor is represented (graphics, sounds, interactive capabilities, etc) should all be consistent with one another.*

    The heuristic was expanded to include examples of "media." The term "consistent" was used as it was felt more precise than the term "agree."

12. **Coherence**

    Original Heuristic *As far as possible make sure that metaphors are coherent with one another.*

    Evaluation Heuristic 8 *The metaphors in the overall user-interface should be as coherent with one another as possible.*

    The heuristic underwent minor rewording. The technical meaning of coherence discussed in chapter 3 was not explained to the evaluators.

13. **Metaphor Influence**

    Original Heuristic *Ensure metaphors do not inhibit access to any functionality.*

    Evaluation Heuristic 9 *The metaphor should not inhibit or prevent access to other functionality available in the user-interface.*

    The heuristic underwent minor rewording.

14. **Non-Metaphorical Aspects**

    Original Heuristic *Always address the aspects of the system* not *explained by metaphors.*

    Evaluation Heuristic Omitted.

    This heuristic was omitted because it was outside the scope of the evaluation. Assessment with this heuristic requires examining the *non-metaphorical* parts of the system, which is a major task.

15. **User Experience**

    Original Heuristic *Harness the user's experience to help identifying useful vehicles.*

    Evaluation Heuristic 10 *The metaphor should not use objects or concepts that are likely to be outside the target user's experience.*

The heuristic was rephrased negatively, to indicate *problems* the evaluators should look for.

16. **Culture**

    Original Heuristic *Design metaphors appropriate to the cultures the interface is intended to be presented to.*

    Evaluation Heuristic 11 *The metaphor should not use objects or concepts that are likely to be inaccessible to a target user's cultural background.*

    The heuristic was rephrased negatively to indicate *problems* the evaluators should look for.

17. **User Types**

    Original Heuristic *Allow expert users to circumvent the metaphorical means of achieving tasks where such circumvention would increase task efficiency.*

    Evaluation Heuristic 12 *The metaphor should provide ways for expert users to circumvent its use if there is a more efficient means of performing a task.*

    The heuristic underwent minor rewording.

18. **Documentation**

    Original Heuristic *Carefully document all structural metaphors used in the interface.*

    Evaluation Heuristic Omitted.

    This heuristic was omitted because the task was considered to be too time-consuming to be practical. Additionally, the inspection of the Project Gallery from chapter 6 indicated that there was very little documentation to work with.

8.2.3 The Metaphors

As discovered in the case study of the taxonomy in chapter 6, there are at least seventeen metaphors to be found in the Project Gallery interface. In order to limit the amount of analysis required of the evaluators, a subset of the known metaphors was selected and specified to the evaluators. Given that evaluation times of roughly one to one-and-a-half hours were expected, eight metaphors were chosen for evaluation. With twelve heuristics to be used, this amounted to ninety-six individual actions required of the evaluators, allowing for approximately one minute per action. While this is a considerable amount of work, the evaluators did generally finish in that time-frame.

Another reason for specifying the metaphors was metaphor identification. The evaluators were not experienced in considering user-interface metaphors in general, and so it seemed unreasonable to ask them to identify metaphors on their own.

Evaluators were presented with the *entire* metaphor, rather than only the *vehicle* (explaining concept). This was done to ensure some uniformity of evaluation.

Finally, a list of representamens in the user-interface for each metaphor was presented to the evaluators. The lists were not complete, but were intended as a guide. This was done because, as already noted, evaluators were not experienced in metaphor identification, and especially finding representamens in the interface.

The metaphors chosen for evaluation were a subset of those in the case study of chapter 6. The sheets with the metaphors on them presented to the evaluators are available in appendix B. The metaphors selected were:

1. THE COLLECTION OF TEMPLATES AND WIZARDS IS A GALLERY
2. THE VIEW OF THE COLLECTION IS A CATALOG
3. THE COLLECTION OF TEMPLATES IS A TOOLBOX
4. THE COLLECTION OF DATA IS A DOCUMENT
5. THE PREFORMATTED DOCUMENT IS A TEMPLATE
6. THE DIALOG BOX(ES) IS A WIZARD
7. THE SOFTWARE IS AN ENTOURAGE
8. THE AREA OF THE WINDOW IS A SCROLL

These metaphors were chosen for various reasons. Most importantly, many were crucial to the Project Gallery itself, rather than being more general user-interface metaphors such as the window metaphor. Additionally, they were generally well represented in the user-interface, or were at least tended to be represented in more than one place, and in more than one manner. Several of the metaphors chosen were key traditional user-interface metaphors, such as the template and document metaphors. Finally, the metaphors were also partially selected with an eye to combining both metaphors the evaluators might already be familiar with in the context of a user-interface (document, template, wizard, scroll), and those that might be unfamiliar (gallery, catalog, toolbox, entourage).

8.2.4 The Evaluation Process

The process given to evaluators is described on the evaluation sheets in appendix B. This involves a brief description of what user-interface metaphors *are*, along with an introduction to the terms "vehicle" and "tenor." Following this is a list of instructions for performing the evaluation, and then a *sample* evaluation of a metaphor (the mail metaphor) from the Project Gallery. The remainder of the evaluation sheets consists of metaphors and representamens to evaluate along with space to write in evaluations. Each evaluator was provided with a copy on paper of these sheets, along with the heuristics from appendix A and was asked to read them and then begin evaluating. As much as possible, the process suggested to evaluators, and followed by the supervisor was kept close to the guidelines of heuristic evaluation developed by Jakob Nielsen [55, 57, 58, 59, 60]. For the remainder of this chapter, only the material from [57] will be explicitly cited, although all of the above references contain valuable information on the evaluation process.

The overall process was to consider each metaphor, testing whether any of the heuristics could be applied to it to identify a problem. Note, however, that as with Nielsen's strategy, "in principle, the evaluators decide on their own how they want to proceed with evaluating the interface." [57, p.158].

The evaluation was performed on an Apple iBook running MacOS X and Microsoft Office X. The initial state presented to the evaluators was of Microsoft Word running with the Project Gallery window on the screen. In each case the evaluators were given the materials in appendices A and B and were asked to begin. The Project Gallery is clearly intended for a broad audience of users, falling under the category of a "walk up and use" interface. This suggested that evaluators could largely be left to their own devices, as in Nielsen [57, p.159].

The evaluation supervisor (the author) sat diagonally behind the evaluator and took notes. It was made clear that asking questions and asking for clarification was desirable, and that these exchanges would be recorded. Importantly, wherever possible, the supervisor did not

offer his *own* opinion on the user-interface or the heuristics, but sought to elicit commentary from the evaluators. This approach follows Nielsen's suggestion that "the responsibility for analyzing the user interface is placed with the evaluator in a heuristics evaluation session." [57, p.157] The evaluators were reminded that the evaluation process was not a test of *them*, but of the interface and the heuristics. Evaluators were informed that the heuristics were prototypes.

The five evaluators had a fair to extensive background in computer science, as well as some knowledge of the process of heuristic evaluation as described by Jakob Nielsen. None of the evaluators had encountered the Project Gallery dialog before, although they all had experience in the use of Microsoft Office software, either on the Windows or Macintosh platform. Not all of the evaluators had experience with the MacOS X operating system, but this was not considered important, as none of the special features of the system are utilised in the Project Gallery, other than standardised widgets. Two of the evaluators also had some specialist experience of user-interface design theory.

Time was not explicitly limited, but evaluators were aware of the expectation that they would take an hour or more in their evaluation. This conforms with Nielsen's suggestion that evaluations take place over one to two hours [57, p.158]. Evaluations took place in several rooms, but all were quiet to help concentration.

As in Nielsen's approach, "the output from using the heuristic evaluation method is a list of usability problems in the interface, annotated with references to those usability principles that were violated by the design in each case in the opinion of the evaluator." [57, p.159] However, in this case, the lists of usability problems were specifically categorised by the metaphors presented for evaluation.

Normally, at the end of a heuristic evaluation session, the group of evaluators are gathered together to discuss their findings and indicate possible ways to correct problems [**?**]. This step was not taken in the present evaluation because it was felt that more insight could be gained by the author analysing the results of the evaluations alone. Additionally, considerable discussion had already taken place with evaluators *during* the evaluations.

8.3 Evaluation Results

Having performed the evaluations, the results were collected together. The format of this section is as follows: first, each of the heuristics provided in appendix A will be presented along with relevant results from the evaluations. The intention here is to examine insights gained into the working of the heuristic when applied by real evaluators. Second, the metaphors and the usability problems discovered are discussed. Here the objective is to indicate how well the heuristics worked in identifying issues with the user-interface. The results will be analysed in some depth in the section 8.4.

8.3.1 The Heuristics

The results of the heuristics will be presented as follows:

- The heuristic itself is reproduced.
- Statistics on how many of the evaluators *used* the heuristic at some point in their evaluation, and how many times the heuristic was applied over *all* evaluations are provided.
- Some discussion is given to the use of the heuristic by the evaluators, along with discussion as to how well the heuristic was applied, based on the author's intention for its use. Additionally, any interesting reaction of the evaluators to the heuristic are noted.

Heuristic 1

The metaphor should not transfer any usability problems from the vehicle (explaining concept) into the user-interface.

Number of evaluators who used it 3

Number of times used overall 4

Overall the heuristic was not applied in the manner expected. The evaluators who did use it tended to use it to complain that functions from the vehicle were *not* applied or applicable in the user-interface.

Heuristic 2

The metaphor should use relevant metonymy (one object standing in for a related object or concept) to represent possible actions in the user-interface.

Number of evaluators who used it 4

Number of times used overall 4

This heuristic was certainly confusing to evaluators. The concept of metonymy was not well understood, for example. Rather than viewing this as "one thing standing for another," evaluators tended to see the heuristic as a means to criticise the general choice of representamens in the interface.

Heuristic 3

Conventional metaphors should always indicate any deviation from the standard functionality or usage of the metaphor.

Number of evaluators who used it 3

Number of times used overall 6

This heuristic was applied fairly closely to the way it was intended. It was still, however, used to identify more general problems with representamens on occasion.

Heuristic 4

The metaphor should use as many useful facts about the vehicle (explaining concept) as is possible without violating other heuristics.

Number of evaluators who used it 2

Number of times used overall 6

This heuristic proved unpopular with the evaluators. Those who did use it, however, did so in the manner intended: as a means of identifying further entailments of the metaphor that *would have been* useful.

Heuristic 5

The metaphor should be as precise as possible to avoid any confusion about its intended meaning.

Number of evaluators who used it 5

Number of times used overall 17

This was the second most used heuristic in the set, and was used by all evaluators. There was some tendency to use this heuristic to identify when metaphors improperly indicated their functionality. Although this is within the heuristic's scope, it is quite general. On the whole, this heuristic identified many useful usability problems.

Heuristic 6

The metaphor should not represent or suggest any functionality that is not accessible.

Number of evaluators who used it 4

Number of times used overall 8

This heuristic was generally applied with great success and accuracy.

Heuristic 7

The ways in which the metaphor is represented (graphics, sounds, interactive capabilities, etc) should all be consistent with one another.

Number of evaluators who used it 3

Number of times used overall 5

This heuristic was used effectively, being applied to those times when the means of representation contradicted each other. On one or two occasions, however, the heuristic was applied to problems where the representamen did not successfully indicate the function, or indicated functions not available. Those problems seem to be covered better by heuristics 5 and 6.

Heuristic 8

The metaphors in the overall user-interface should be as coherent with one another as possible.

Number of evaluators who used it 5

Number of times used overall 18

This was the most used heuristic in the set, and was used by *all* evaluators. The heuristic was often used in the manner intended, but was also quite often used as a catch-all for problems felt to exist between a metaphor and other aspects of the interface, which were not necessarily metaphorical. The incoherences did not tend to involve conflicting metaphorical entailments, but rather a more general confusion generated by the interaction of two metaphors. Another issue that arose was that several of the incoherences identified may not have been very serious. The question of how bad an incoherence must be before being a problem was raised by this.

Heuristic 9

The metaphor should not inhibit or prevent access to other functionality available in the user-interface.

Number of evaluators who used it 2

Number of times used overall 3

This heuristic was largely unused, although when it was applied it was successful in identifying usability problems.

Heuristic 10

The metaphor should not use objects or concepts that are likely to be outside the target user's experience.

Number of evaluators who used it 5

Number of times used overall 9

This metaphor was used by all evaluators, and was applied as intended.

Heuristic 11

The metaphor should not use objects or concepts that are likely to be inaccessible to a target user's cultural background.

Number of evaluators who used it 4

Number of times used overall 6

As above, the culture heuristic was applied as intended.

Heuristic 12

The metaphor should provide ways for expert users to circumvent its use if there is a more efficient means of performing a task.

Number of evaluators who used it 0

Number of times used overall 0

This heuristic was *never* applied to the interface. Whether this is because the interface did not contain any instances of violation of the heuristic, or because it was hard to understand is not completely clear. Note, however, that the project gallery *is* easily dismissed, which leads expert users directly into the program they wish to use. This supports the idea that the heuristic was simply not applicable in the Project Gallery.

8.3.2 The Metaphors

In this section the metaphors used in the evaluation will be presented. For each metaphor various results of the evaluative process will be given:

- The number of unique problems found by all evaluators for the metaphor
- The numbers of the heuristics most violated by that metaphor. "Most violated" is taken to mean that three or more (a majority) of evaluators cited that heuristic as a problem for the metaphor.
- Discussion of useful insight gained from the evaluations to demonstrate that the heuristics *did* promote productive evaluation.

THE COLLECTION OF TEMPLATES AND WIZARDS IS A GALLERY

Number of unique problems found 16

Heuristics most violated 5, 8

A key issue identified with the Project Gallery was that it was not like a real gallery in enough ways (heuristic 5). Evaluators complained, for example, that it did not reflect the fact you can generally take in everything at a glance in a gallery, that you can move closer to inspect things, and that the metaphor did not discourage interpreting the contents as art well enough. Additionally, evaluators used heuristic 8 to suggest that the metaphors inside the Project Gallery did not really fit with each other, such as the presence of wizards and toolboxes.

THE VIEW OF THE COLLECTION IS A CATALOG

Number of unique problems found 11

Heuristics most violated 5, 8

An interesting point raised under heuristic 8 was the confusion between the catalog metaphor used for viewing, and the category of "catalogs and menus" in the categories pane of the user-interface. A key issue raised, using heuristic 8 once more, was that the catalog and gallery metaphors were felt to conflict with each because they seemed to explain the *same* functionality in the interface. Evaluators felt this could be confusing to users. One evaluator pointed out, using heuristic 5, that catalogs generally contain far more information than is given in the Project Gallery catalog. This issue was extended further with reference to heuristic 9 by noting that there is *less* information available in the catalog view than in the list view, suggesting the metaphor inhibits access to some information. Another issue raised was that the combination of a room-based metaphor (gallery) and a book-based metaphor (catalog) was confusing. Although this was listed as violating heuristic 2 instead of heuristic 8, the insight is important.

THE COLLECTION OF TEMPLATES IS A TOOLBOX

Number of unique problems found 18

Heuristics most violated 5, 6, 8

Most evaluators suggested that the toolbox metaphor was extremely imprecise. They felt that there was no real suggestion of the way in which the interface element functioned as a toolbox, especially since the templates contained within it were not really *tools* per se. Evaluators also felt this lead to a violation of heuristic 6, as the notion of a toolbox promised functionality that did not seem present in the interface, such as actual writing *tools* such as pencils and so forth, as well as the ability to *open* it. Two evaluators were confused by the presence of a toolbox in a gallery (heuristic 8), and one was also puzzled by the toolbox being the *only* one in the interface. One evaluator noted that the tool box was "identical in functionality to other categories, yet it suggests it's a toolbox rather than a category." One evaluator also complained that the yellow "blob shapes" which appeared along with the explanation of the Project Gallery when the toolbox was selected were confusing and lead them to miss seeing the actual functionality of the toolbox. The assumption was that the "blobs" were part of the metaphor, or another metaphor that was misunderstood. In fact, the "blobs" are simply a decorative part of the interface so it is a concern that they were interpreted as metaphorical. As can be seen from the problem count, the toolbox was by far the most problematic metaphor for the evaluators.

THE COLLECTION OF DATA IS A DOCUMENT

Number of unique problems found 12

Heuristics most violated 8

Two of the evaluators raised the coherence issue (heuristic 8) of the similarity between the icons for a "Web Page" and a "Word Document." The issue for the evaluators was that the "Web Page" icon had a curled over corner, suggesting it was multi-page, while the "Word Document" did not. This was felt to be confusing and unjustified. In a similar vein, another evaluator felt that there were coherence issues between the various icons for the "Blank Documents" category which differed greatly in their representations. Another interesting observation made by two evaluators under heuristic 7 was that the representation of the "Blank" Word Document already appeared to have a large amount of writing on it, as well as a large "a" printed over the middle of the paper. It was suggested this might alarm users as to what to expect when using the document. This concern was mirrored, though under heuristic 5, with the issue that the document icon possibly indicated that the document "will have a header of some sort." Also under heuristic 5 was the suggestion that the "Mail Message" option was in the wrong category because a mail message might not seem like a document to many users.

THE PREFORMATTED DOCUMENT IS A TEMPLATE

Number of unique problems found 10

Heuristics most violated n/a

The template metaphor suggested a wide range of problems to the evaluators, and led them to cite many different heuristics to describe them. Two evaluators felt that the template metaphor was uncomfortable in the overall gallery metaphor, suggesting that the gallery metaphor was perhaps inappropriate to describe a collection of templates. Other evaluators felt the representation of templates was so similar to documents that they would be hard to differentiate for new users (heuristics 6 and 9 were cited here).

THE DIALOG BOX(ES) IS A WIZARD

Number of unique problems found 12

Heuristics most violated 10, 11

As might be expected, the most common problems found with the wizard metaphor concerned the user's general and cultural background. A majority of evaluators commented that wizards could be interpreted in a large number of ways, and thus could cause problems for different users depending on their background. Additionally, it was suggested that the idea of a wizard would not necessarily work across cultures and could even be offensive to some cultural backgrounds. Another issue identified by two evaluators under heuristic 8 was that the Envelope Wizard's icon did not include an image of an *envelope* on it, even though the Mail Message icon did. This was felt to be an incoherence, although perhaps it is better described as an inconsistency between representations. An important observation was that the metaphor violates heuristic 4 in that it entirely omits the concept of *person-hood* from the wizard metaphor. As the evaluator noted, the "UI wizard has no personal representation at all." This lack of personality was noted by another evaluator under heuristic 6. Additionally, the same evaluator noted under heuristic 4 that the wizard icons "aren't very graphically descriptive about what they do. e.g. Envelope Wizard looks identical to Letter Wizard."

THE SOFTWARE IS AN ENTOURAGE

Number of unique problems found 8

Heuristics most violated 10

The most common problem suggested with the entourage metaphor was simply that users might well have no idea as to what the word itself meant (heuristic 10). If the user cannot come to terms with the *meaning* of the metaphor, it is rendered useless to them. Other problems suggested were that the name did not seem to convey the underlying function (heuristics 5 and 6 were cited), and that the concept of an entourage seemed overly similar to the wizard metaphor (heuristic 8). A final, and interesting, problem was that one evaluator felt as though an "Entourage Document" would be a document which followed the user around in some way.

THE AREA OF THE WINDOW IS A SCROLL

Number of unique problems found 5

Heuristics most violated n/a

Evaluators all found the scroll metaphor extremely difficult to evaluate because it was felt to be *too* familiar to them. The general argument against treating it as a metaphor was that almost all users of the system were likely to have their only experience of "scrolling" on a *computer*, rather than with an actual scroll. The problems found were very specific and usually involved citing heuristics 10 or 11 and suggesting the concept of a scroll was unfamiliar to users.

8.4 Analysis of Results

This section is intended to collect together the many insights that were obtained over the course of the evaluations, and in examining the collected results. Although much valuable

information was gathered on the interface itself, the focus of this section will be on the analysis of the heuristics and the evaluation process. The Project Gallery will be briefly discussed, however, before a more detailed look at what was found out about the process and heuristics.

8.4.1 The Project Gallery

Perhaps the most useful insights concerned issues of incoherence between the metaphors presented, and issues of representation of functionality by metaphors, either present or not. Overall, evaluators felt the metaphors in the Project Gallery were implemented in too *superficial* a manner, with no genuine substance to them.

One of the most interesting points made by the evaluators was the conflict between the gallery and catalog metaphors. In general, evaluators felt that the gallery metaphor dominated the catalog metaphor, because it was the one encountered by them first (in the title of the window, for example). Thus, by the time they came upon the catalog metaphor, most of the functionality of the window had already been attributed to the gallery metaphor. As one evaluator said, "both the catalog vehicle and the gallery vehicle are leveraging the same real-world knowledge."

Another useful insight concerned the strange nature of the "blobs" visible in the Project Gallery explanation which appeared when a super-category was selected. One evaluator felt that these blobs comprised part of a metaphor they did not understand and was very distracted by them. Although the blobs are almost certainly only decorative, their distracting nature is a problem.

Most evaluators suggested that "Entourage" was an unhelpful name to use for the software because users, including themselves, would either not know the word, or be unable to connect its meaning to the user-interface. Although, as discussed in chapter 6, it does seem the entourage metaphor has some useful meaning, the evaluators generally did not support its use.

Although the problems identified are not fatal to the operation of the Project Gallery it is important to note they are likely to be a hindrance. This is emphasised by the fact that the Project Gallery is clearly intended to be very usable, and is the basic interface to Microsoft Office X for all new users. The Project Gallery option is the first available in the "File" menu, which is traditionally the most used menu in programs in the operating system. The number of usability problems identified in only a small number of its metaphors, therefore, is certainly a usability concern.

8.4.2 The Process

On average, evaluators took one and a quarter hours to complete the evaluation of the Project Gallery. In this time, evaluators found an average of 18.4 usability problems based on the heuristics and metaphors prescribed.

As noted in section 8.2 and also reflected in the selection of heuristics and metaphors, several major decisions were made in the design of this evaluation. Some of these decisions on process will now be discussed in light of the insights gained by working with the group of evaluators and their findings.

Metaphor Selection

In the evaluations a series of eight metaphors were specifically chosen for the evaluators. This did lead to some confusion, and at least two evaluators asked directly whether they were supposed to use those metaphors as a general or strict guide in the assessment. The defined set of metaphors certainly does not match Nielsen's standard for evaluation that,

"in principle, the evaluators decide on their own how they want to proceed with evaluating the interface." [57, p.158]. Although the evaluators had every freedom beyond selecting the metaphors to evaluate, this restriction did have serious implications. Most importantly, it did not facilitate the usual approach to user-interface evaluation, which is to work through the interface as a *whole*, finding problems where they occur, and then to cite heuristics these problems violate. This process was not possible with prescribed metaphors and may have lead to too narrow a focus in the evaluators' minds when thinking about the interface. This is perhaps evident in the suggestions that the gallery metaphor dominates the catalog metaphor: the gallery metaphor is suggested for evaluation *before* the catalog metaphor on the evaluation sheets.

The necessity of pre-identifying metaphors is removed if evaluators can be taught to identify them themselves. This would lead to a more natural evaluation process in general. It would be far less prescriptive, requiring only the heuristics and the interface. It does not seem that the process of metaphor identification would be overly difficult to teach, and so this is certainly a next step in the evolution of metaphor usability evaluation.

Representamen Selection

As discussed in section 8.2.3 and shown in appendix B the evaluators were presented with a bulleted list of representamens for each metaphor. Although this list was not intended as complete, several evaluators either assumed or asked whether it was. At least one evaluator noted she was relieved the representamens bullets were listed because she doubted she would see the metaphors in the user-interface as she was so used to them. Another interesting reaction was that one evaluator *numbered* the bullet points so that he could refer to the particular aspects of the representamen in his comments. Future versions of the study should certainly add this feature.

Overall it seems that while it appears unadvisable to leave the representamens out, it is also problematic to have an incomplete list. The concern is that an incomplete list might cause evaluators to *miss* other aspects of the representamen in the user-interface. The solution appears to be to provide a numbered list of the locations of *every* instance of representation of the metaphor in the user-interface, although this may be large.

Another important fact about the representamens is that they are the *only* means of perceiving the metaphor in the user-interface, by definition (see chapter 4). Given this, it may be the case that the focus of the heuristic evaluations could be moved to specifically assessing the *representamens* of metaphor in the system, and the metaphors *through* them.

Note that the representamen identification is only required for relatively inexperienced metaphor evaluators. One of the key skills to be gained by evaluators in this area is the ability to identify the representamens of metaphors independently. This ties in closely with the idea of teaching metaphor identification discussed in the previous section and, in fact, the two are likely inseparable.

The Suggested Approach

In the evaluation sheets provided to the evaluators, a general approach was suggested:

> As a general guide: for each heuristic pause to think about what you feel the metaphor ought to mean in a user-interface. Then examine the actual presentation in the interface (pointers to this are given with the metaphors). Finally, look at the list of heuristics and establish whether any of them have been violated.

This provided a well-structured approach for evaluators, which was necessary given their inexperience with the heuristics. The approach did, however, limit the freedom the evaluators

had when examining the user-interface. As already discussed in section 8.4.2 it would be desirable to train evaluators to identify metaphors and representamen on their own. At least one evaluator felt that the chosen approach was counter-intuitive. Most particularly, this was because the process of methodically applying heuristics to a metaphor did not seem natural to evaluators: generally, evaluations are problem-focused rather than heuristic focused.

Other Insights

Some other useful information yielded during the evaluations concerned various language corrections for the evaluation sheets. Two specific examples are the suggestion that the language used when discussing representamens for the Word Document should be the "*label* of the Word Document," rather than its "*title*." Another evaluator suggested that the language used when explaining how to find the explanation material on the Project Gallery was confusing. This is quite possibly, however, because the actual *act* of finding this information is confusing.

Another question raised by the evaluation was the issue of how *fussy* evaluators ought to be. Although evaluators were requested to use severity ratings, they more often than not omitted them. Thus, some of the problems identified did not seem particularly serious, but there was no way of classifying them. In general, it seems that a strict adherence to the severity rating system would solve this problem if combined with a greater awareness of how much metaphor usability problems affect the user. This will be discovered by further testing, as discussed in section 8.5. Additionally, Jakob Nielsen has written an short essay specifically concerning the nature of severity ratings which might help evaluators to more confidently use them [61].

One of the more important insights was the confirmation of the idea that metaphors can become so conventional as to become definitive. Three evaluators felt that the scroll metaphor was too familiar to be considered metaphorical any more, and one felt this applied to the template metaphor, which formed her only true experience of templates. It seems clear that this will continue to happen as people become more likely to use a user-interface metaphor's representation of a vehicle than the real vehicle in the world.

8.4.3 The Heuristics

The most important insights gained from the evaluation concerned the heuristics themselves. The evaluations were performed *specifically* to road-test them and the long-term objective is to develop a set of extremely usable and definitive heuristics for the usability of user-interface metaphors.

One immediate issue that was raised by evaluators was that there were simply *too many* heuristics. This seems reasonable, and perhaps the number of heuristics should be reduced to within the chunking rule of 7 ± 2 as suggested by G. A. Miller [49]. An interesting related insight was that one evaluator went through the heuristics and *paraphrased* them according to his interpretation. This was perhaps done to reduce the memory load, and also suggests that the wording of the heuristics still needs work to be immediately accessible.

Perhaps the most important insights, however, concerned specific metaphors in the set provided, and even metaphors that *should* have been in the set provided. In the following sections, knowledge gained about the individual heuristics will be put forward.

Related Heuristics

Several evaluators explicitly identified relationships between the heuristics:

Heuristics 2 and 6 One evaluator felt that heuristics 2 and 6 were effectively converses of each other and so should be grouped or combined into one heuristic. This comment likely stems from the general misunderstanding of metonymy in the evaluations. Metonymy was probably too technical a concept for evaluators who had not read chapter 3. That said, the heuristics do deal with the representation or non-representation of functionality in the system, and can be said to be related in that way. The general heuristic which might capture the real intention of the evaluators is "do represent functionality present in the system, and don't represent functionality not present in the system."

Heuristics 4 and 5 Two evaluators felt that these heuristics were very similar and hard to differentiate. Note that heuristic 5 was used by all evaluators, while 4 was only used by 2. This suggests that evaluators felt they could more easily apply the language used in 5, than the similar heuristic in 4. On reflection it *does* seem that the two heuristics are very similar, and could be combined into a single heuristic.

Heuristics 5 and 6 One evaluator felt that heuristics 5 and 6 were similar to each other in meaning. Although this interpretation is valid, it does not seem that the heuristics ought to be combined in this case. Heuristic 5 concerns the specificity of the metaphor, while heuristic 6 is more concerned more generally with functionality.

Heuristics 10 and 11 Most evaluators felt that heuristics 10 and 11 were too similar to apply independently. It was recommended they were combined into a single heuristic concerning user-experience, and which acknowledged culture as a key part of that. This seems very reasonable, but the explicit reminder in heuristic 11 to consider culture as well as general experience still appears valuable.

Missing Heuristics

On several occasions evaluators suggested they could have used an additional heuristic. This most specifically relates to a heuristic cautioning against *redundant* metaphors in the user-interface, commonly raised as a problem with either the gallery or catalog metaphors.

Another concern some evaluators had was the issue of distinguishing between problems relating to a metaphor's *own* functionality, and those relating to *other* functionality in the user interface. Although it seems that heuristics such as 5, 6 and 9 do cover that concern, this type of problem may need to be made more explicit.

One evaluator suggested that it would be useful to have a heuristic which commented on the general *mis*use of metaphors, or the selection of the "wrong" metaphor for a task. Although this is rather general, it might prove a valuable heuristic, which could, in turn, reference *other* heuristics in the set, to justify its citation.

A final issue, raised by several evaluators, was the need for a heuristic concerning the *appropriateness* of a metaphor's representamen in the user-interface. Although heuristic 7 does deal specifically with the representamen, it concerns co-ordination, rather than appropriateness. There may well be room, therefore, for such a heuristic.

Heuristics Language

The language used in the heuristics presented to the evaluators did undergo several edits, but the heuristics can always be made more simple and easier to understand. As an example, consider the suggestion of one evaluator that heuristic 1 be rephrased along the lines of "*Don't use something that is already hard to use as a metaphor.*" This is far more direct and also

avoids the technical use of "vehicle" in the heuristic. This sort of rephrasing can probably be applied to *most* of the heuristics.

Misunderstandings

Heuristic 2 tended to generate some misunderstanding, largely because it used the technical term "metonymy," which evaluators were not necessarily familiar with. Although most of the evaluators used the heuristic at some point, they generally didn't seem to grasp the intended use of it. This suggests the heuristic is probably over-specific, and either evaluators need training in the concept of metonymy, or the heuristic itself needs to be simplified to remove the concept.

There was some small confusion as to what a "conventional" metaphor was. It seemed as though evaluators used heuristic 3 to cite anything generally out of the ordinary, rather than to focus on metaphors that were supposedly well established in user-interface design. Once again, training would solve this problem.

The rest of the heuristics were largely understood and applied successfully to the Project Gallery. Although heuristic 12 was *never* applied, it seems that evaluators did understand, but simply did not see any problems which related to it.

Most Useful Heuristics

Looking at the statistics from section 8.3.1 it is clear that heuristics 5 and 8 were found to be the most useful. Evaluators who used the heuristics applied them more than three times each on average. Heuristic 5 was probably found useful because of its use of the term "confusion." This enabled evaluators to apply it to situations where the use of metaphor seemed unusual or vague. Heuristic 8 seems to have been successful because it allowed evaluators to *compare* metaphors in the user-interface, although one evaluator suggested the need to identify other metaphors because of this was troublesome. Heuristic 10 was also widely applied, and this is probably because it allowed evaluators to draw on their own experience of metaphors to judge how reasonable they seemed.

Least Useful Heuristics

The least useful heuristic according to the study was heuristic 12, which was never used. As already noted, however, this was probably a function of the success of the user-interface. Heuristic 9 was very rarely applied, but was still successful at identifying important issues, such as that the catalog view of the gallery prevented access to an important piece of information (file type).

The Use of Examples

A final point concerns the apparent success of the examples. One evaluator went so far as to *identify* heuristics by their example. For example, the evaluator referred to "the flushing toilet one" to convey heuristic 7. Other evaluators commented that the examples helped to clarify the heuristics considerably. Although this clearly indicates that the heuristics need more work to make them more quickly intelligible, the idea of a "canonical" example for each heuristic has merit.

8.5 Further Work

This final section discusses further work both to improve the heuristics and the process for testing them based on the results and analysis of the heuristic evaluation.

Redevelop the Heuristics The heuristics need much more work to make them easy to understand. Much insight was obtained from the evaluators, and directions for improvement have already been suggested.

Train the Evaluators While this would require a considerably larger investment in time, it is likely that this investment would pay off with greater understanding and even better analysis of the interface. Developing the materials for training evaluators would be a substantial task. The training might include:

- Evaluators learning the basic theories which ground the heuristics.
- Evaluators learning how to identify metaphors and their representamens in a user-interface, to give them more freedom during the evaluation.
- Evaluators learning the heuristics and becoming familiar with them prior to the evaluation. This would reduce dependence on the written listing and give more focus to the actual interface.

Give the Evaluators More Freedom Trained evaluators would not require a specific list of metaphors and representamens. Instead, they could be given the heuristics and an interface, and be asked to apply one to the other in a style much closer to Nielsen's approach [57].

Redevelop the Interface It is perfectly possible to evaluate an interface which can then be *improved* by fixing the problems identified by evaluators. By performing two or more iterations of evaluation followed by amending the interface, results on whether the heuristics lead to the ability to directly *improve* the interface could be obtained.

Test Metaphor Effect on Usability As above, if the heuristics were used to redevelop an interface iteratively, Nielsen's heuristics could be applied along the way to test overall usability. If the usability improved via changes based only on the metaphor heuristics, there would be a strong empirical suggestion that metaphors form an important part of usability.

Use the Heuristics During Design Although the heuristics have been shown to be successful evaluatively, it is desirable to test their influence on the design process. This could be done with a small software project requiring a user-interface involving metaphors. Nielsen's heuristics could be applied similarly, to test the effect of the metaphor heuristics on usability, as above.

8.6 Conclusions

As has been shown in the analysis section (8.4), a considerable amount of insight into the heuristics and the process of heuristically evaluating user-interface metaphors has been gained. The heuristics have been shown applicable to a real world user-interface and have successfully identified usability problems. Furthermore, the suggestions in section 8.5 indicate that this study has generated proposals for considerable future work. This will chiefly involve the enhancement and extension of the heuristics developed in chapter 7 as well as the general concept of evaluating interfaces from the specialised position of metaphor analysis.

Chapter 9

Conclusions

The overall aim of this thesis has been to explore the nature of user-interface metaphors. In particular, the aim was to draw from theory from outside the domain of computer science in order to create a detailed account of this interface design technique. Linguistic philosophy, in the form of Lakoff and Johnson's work *Metaphors We Live By*, and the semiotics of Charles Peirce and Umberto Eco have served as the theoretical basis for the research undertaken.

This thesis covers a large amount of material in three main areas: the identification of the issues and the history of user-interface metaphor; a proposed theoretical solution to the issues; and practical applications of the research in the thesis to demonstrate its usefulness. These three major contributions will now be discussed in more detail.

Chapter 2 focused on examining typical definitions of user-interface metaphor and then discussed a large amount of the current work in the area. The chapter positioned the significance of metaphor in the understanding of computers and suggested it was a very important component. It was noted, after examination of research into user-interface metaphors that the usual response has been to offer common-sense and practical approaches, particularly in the form of heuristics. It was pointed out that the most obviously lacking research was into the *concept* of user-interface metaphor itself. Most previous work has proceeded directly to considerations of application, rather than understanding.

Having identified a crucial issue in the research on user-interface metaphor, a possible approach to a solution was offered in chapters 3 and 4. In chapter 3 the taxonomic work of Lakoff and Johnson was applied to metaphor in the user-interface. This provided a means of specifically classifying different kinds of user-interface metaphor. Additionally, insight from Lakoff and Johnson on the working and structure of metaphor was discussed, particularly the notion of metaphorical entailments. The classification of metaphors was also claimed to aid in understanding the particular ways in which metaphor can be used in the interface.

Chapter 4 followed this relatively high-level classification with a detailed examination of the underlying structure of the individual user-interface metaphor. This was performed by first explaining the discipline of semiotics, and specifically the approach of Charles Sanders Peirce, and then applying a Peircean triadic model of the sign to metaphor in the interface. The chapter demonstrated how this model is able to serve both as a model of a particular metaphor, as well as describing the design process to some degree.

Together, chapters 3 and 4 have provided a detailed examination of the concept of user-interface metaphor at different levels. These findings are intended as a solution to the problem of understanding user-interface metaphor more deeply.

Chapter 5 reiterated the information from chapters 3 and 4 in a concise and structured manner, focusing on the major items of terminology which were introduced. This chapter is intended to serve as a quick reference point for the concepts contained in the theoretical research.

The final component of the thesis concerned the practical applications of the theories developed. Chapter 6 provided a detailed example of the application of the concepts from chapters 3 and 4 to a real user-interface. Metaphors in the user-interface were classified according to the material derived from Lakoff and Johnson, and then analysed according to this classification. A selected set of metaphors were then examined in detail according to the semiotic model provided in chapter 4. The application of the semiotic model demonstrated its usefulness in at least two major ways: as a technical vocabulary for the more exact discussion of user-interface metaphors; and as a structured and componential view of such concepts. The overall impression given by the case study was that the analytic and classificatory tools developed in the thesis are highly applicable to existing user-interfaces, and ought to be equally useful for employment *during* the design and implementation process. One clear direction for future research is to build a user-interface using the guidance of the taxonomy.

Chapter 7 presented a set of usability heuristics largely derived from the material in chapters 3 and 4. The heuristics are generally intended for informing the design process, rather than the evaluation of completed interfaces. It was demonstrated in chapter 8, however, that a subset of the heuristics *can* be used to evaluate existing user-interfaces. The heuristics elicited considerable insight into the workings of metaphors in a user-interface. Chapter 8 also sought to find ways to improve both the heuristics and the process of using them and did so with substantial success.

Having indicated the structure of the thesis and its intentions, the overall contributions of the research can now be indicated in order to conclude:

1. The application of Lakoff and Johnson's research to user-interface metaphor in chapter 3 provides a structured means of classifying metaphor in the interface. It additionally sheds light on the fact that different *kinds* of metaphor are used for different purposes and provides explicit guidance.

2. The semiotic model developed in chapter 4 allows a far deeper and componential view of user-interface metaphors. Most of all, this will allow a more comprehensive understanding of how user-interface metaphors function in detail. Additionally, it has been suggested that the semiotic model also provides a description of the design process from metaphor conception to user-testing.

3. In addition to the above, the demonstration of a detailed application of semiotics to user-interface metaphors indicates that other semiotic techniques may also be transferable. Semiotics as a whole has been applied to many other forms of media and, via the semiotic model provided here, insights from these other domains may well be applicable to user-interface metaphor.

4. An important contribution of the overall taxonomic approach established in chapters 3 and 4, and collected together in chapter 5 is the introduction of an extensive vocabulary and conceptual framework for the discussion of user-interface metaphor. It is hoped that this will lead to more understanding between designers and academics when researching and applying the concept of metaphor use in the interface.

5. Finally, the heuristics developed in chapter 7 have been shown to be useful in evaluating user-interfaces. The heuristic evaluation in chapter 8 showed in considerable detail that the heuristics were successful at identifying issues in the metaphors of a user-interface. The heuristics emerge from a solid theoretical basis, and so are justified on those grounds also. Further practical work with the heuristics, as suggested in chapter 8 is an important direction for future research.

The above list demonstrates that this thesis forms a strong contribution to user-interface metaphor research, and through this to human-computer interaction research in general. The thesis is well balanced between detailed theoretical research based on linguistic philosophy and semiotics, and also empirical studies of the concepts that show the ideas are applicable in practice. Ultimately, this thesis has provided a detailed and demonstrably practical account of user-interface metaphors.

Appendix A

The Heuristics

1. *The metaphor should not transfer any usability problems from the vehicle (explaining concept) into the user-interface.*

   Example: Using a vehicle of a two-button digital watch with 10 functions mapped to them would be a mistake. This will simply transfer the problems already inherent in the watch to the user-interface.

2. *The metaphor should use relevant metonymy (one object standing in for a related object or concept) to represent possible actions in the user-interface.*

   Example: A pair of scissors is used to indicate the "cut" function in Microsoft word, which is a relevant use. It would not be relevant to indicate the "cut" action with an image of a sharp stone.

3. *Conventional metaphors should always indicate any deviation from the standard functionality or usage of the metaphor.*

   Example: If a new Operating System had a trashcan that *automatically* deleted files put in it, this would need to be carefully indicated. Normally the user must explicitly indicate they want the files in the trash deleted.

4. *The metaphor should use as many useful facts about the vehicle (explaining concept) as is possible without violating other heuristics.*

   Example: Social conventions such as politeness are crucial when dealing with people. Thus a "person" metaphor should include this social aspect of people.

5. *The metaphor should be as precise as possible to avoid any confusion about its intended meaning.*

   Example: There is a different level of specificity between a "document" metaphor and an "office document" metaphor. The more specific a metaphor is, the more the user will know the exact intention.

6. *The metaphor should not represent or suggest any functionality that is not accessible.*

   Example: A book metaphor might present the screen as a book, but there should not be a curled over corner of the page if this cannot be used to turn that page.

7. *The ways in which the metaphor is represented (graphics, sounds, interactive capabilities, etc) should all be consistent with one another.*

   Example: If, when emptying the trash, the sound of a flushing toilet is made, there is an inconsistency in the representation.

8. *The metaphors in the overall user-interface should be as coherent with one another as possible.*

   Example: Explaining application launching in the desktop metaphor with a metaphor of chemical reaction creates an incoherence. Those two vehicles simply do not go together.

9. *The metaphor should not inhibit or prevent access to other functionality available in the user-interface.*

   Example: The trashcan metaphor in earlier versions of MacOS was problematic because it was also used for ejecting disks (by moving them to the trash). The disk-ejection function was inhibited by the trashcan metaphor as users were frightened of "throwing away" their disks.

10. *The metaphor should not use objects or concepts that are likely to be outside the target user's experience.*

    Example: Using a metaphor derived from complex nanotechnology research will prove inaccessible to the average user.

11. *The metaphor should not use objects or concepts that are likely to be inaccessible to a target user's cultural background.*

    Example: Using a metaphor concerning American Football will be inaccessible to users from cultures where that sport is not played. Similarly, a metaphor concerning the subtleties of Buddhism will likely prove incomprehensible to users who do not practice that religion.

12. *The metaphor should provide ways for expert users to circumvent its use if there is a more efficient means of performing a task.*

    Example: Forcing all users to delete files by moving them to the trash manually, one at a time, will be annoyingly inefficient to expert users. Instead, providing a hot-key, and the ability to delete multiple files is desirable.

Appendix B

Heuristic Evaluation of the Project Gallery

Thank you for taking the time to take part in this heuristic evaluation of the Project Gallery in Microsoft Office.

What Is A User-Interface Metaphor?

What is a metaphor?

A metaphor uses a *vehicle* to explain a *tenor*. For example:

Metaphor JULIET IS THE SUN

Vehicle/Explaining Concept THE SUN

Tenor/Concept to be explained JULIET

Juliet (the tenor) is explained by identification with the sun (the vehicle). So she is "bright", "warm," "life-giving," "high above us," and so on.

What is a user-interface metaphor?

In a user-interface metaphor, the *vehicle* is normally some real world object or concept, and the *tenor* is some part of the user-interface which needs explaining. For example:

Metaphor FILE DELETION IS USING A TRASHCAN

Vehicle/Real world concept USING A TRASHCAN

Tenor/System concept FILE DELETION

File deletion (the tenor) is explained by identification with use of a trashcan (the vehicle). The user removes files by "putting them in the trash," retrieves files by "taking them out of the trash" and finally deletes them by "emptying the trash."

The Evaluation Process

- Read the TWELVE (12) heuristics provided to initially familiarise yourself with their content.
- If there is a heuristic which does not make sense to you, please make a note of it, or tell the evaluation supervisor.
- Have a look at the sample evaluation that follows this section for an idea of the kinds of things you might find and write about.
- Examine each of the EIGHT (8) metaphors listed on the evaluation pages according to the heuristics and note down whenever a heuristic is violated, along with a rating of how serious you think the problem is from 1 to 5 (5 being the *most* severe).
- As a general guide: for each heuristic pause to think about what you feel the metaphor ought to mean in a user-interface. Then examine the actual presentation in the interface (pointers to this are given with the metaphors). Finally, look at the list of heuristics and establish whether any of them have been violated.
- If you have time to make a second pass through the metaphors, please do, although this is not required.
- Feel free to interact with the interface to discover anything you wish to know about the metaphors you are evaluating.
- If you dismiss the Project Gallery window for any reason you can get it back by selecting it in the "File" menu of a running Microsoft Office program.
- If you find a problem in the user-interface that seems related to metaphors, but you cannot see a heuristic you think applies, please note it down anyway. This will help in the development of new, better heuristics.

A Sample Evaluation

THE WAY OF SENDING DATA IS MAILING

- In "Blank Documents" category: title of "*Mail* Message."
- Graphic of envelope for "Mail Message" icon.

Heuristic 5 is partially violated because the mail metaphor has not been sufficiently distinguished from the real world task of mailing which is presented in the "Letters-Envelopes" category. This may lead a user to think that the "Mail Message" option is actually for writing a *physical* letter.

It could be argued that the image of an envelope violates **heuristic 6**, because there is no functionality which involves the use of an envelope in the mail metaphor. Envelopes just don't feature, yet one is depicted. This may fall under **heuristic 2** also, because it is a semi-irrelevant metonymy, given its inapplicability in the actual metaphor.

There may be a violation of **heuristic 9**, because the presence of the "Mail Message" option, which is one of the *first* things a user sees, may lead them to think they can write a *letter* with it. This will lead them away from using the templates in the "Letters-Envelopes" category, which are the tools actually provided for letter writing.

1. **THE COLLECTION OF TEMPLATES AND WIZARDS IS A GALLERY**

- Title of main window ("Project Gallery").
- The text seen when selecting "Business Forms" or "Home Essentials," for example.
- Presence of the 'catalog' metaphor.

2. **THE VIEW OF THE COLLECTION IS A CATALOG**

- Drop-down "View:" metaphor at bottom left.
- Visual layout of right-hand pane when displaying templates and wizards available in a category.

3. **THE COLLECTION OF TEMPLATES IS A TOOLBOX**

- Title of "Writing *Toolbox*" category in left-hand pane.

4. **THE COLLECTION OF DATA IS A DOCUMENT**

- Title of "Blank *Documents*" category in left-hand pane.
- Title of "Word *Document*" in that category.
- Icon indicating documents is a sheet of paper.

5. **THE PREFORMATTED DOCUMENT IS A TEMPLATE**

- Title of "My *Templates*" category in left-hand pane.
- In explanation of Project Gallery display in right-hand pane when a category with no direct templates is selected (such as "Business Forms").
- In many categories, such as "Newsletters," the contents are all templates.
- In "Create:" menu at bottom right.

6. **THE DIALOG BOX(ES) IS A WIZARD**

- In "Letters-Envelopes" category in title of "Envelope *Wizard*."
- In "Blank Documents" category in title of "List *Wizard*."
- Other wizards in the Project Gallery.
- Graphic of "magic wand" in standard icon for the wizards.

7. **THE SOFTWARE IS AN ENTOURAGE**

- In "Show:" menu at the bottom of the window in the center.
- In the stylised "e" shown on the Entourage Documents in the "Blank Documents" category: "Mail Message" and "Calendar Event."

8. **THE AREA OF THE WINDOW IS A SCROLL**

- The *scroll*-bar attached to the left-hand pane. (Appears when there is more information than fits a window.)

Bibliography

[1] Tamara Adlin, Holly M. Jamesen, and John Pruitt. Personas: Exploring the real benefits of imaginary people. `http://www.chiplace.org/techniques/show-article.jsp?id=1`, 2002.

[2] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, 1977.

[3] Peter Bøgh Andersen. A semiotic approach to programming. In *The Computer as Medium*, pages 16–67. Cambridge University Press, 1993.

[4] Peter Bøgh Andersen. *A Theory of Computer Semiotics.* Cambridge Series on Human-Computer Interaction. Cambridge University Press, 1997.

[5] Peter Bøgh Andersen. What semiotics can and cannot do for HCI. In CHI'2000 Workshop on Semiotic Approaches to User Interface Design., 2000.

[6] Laehyun Kim Anna. An implicit-based haptic rendering technique. `http://citeseer.nj.nec.com/527104.html`.

[7] Apple Computer, Inc. Staff. *Macintosh Human Interface Guidelines.* Addison-Wesley, 1992.

[8] Aristotle. *Poetics.* William Heinemann, 1927. W. H. Fyfe (translator).

[9] Pippin Barr, Robert Biddle, and James Noble. Icons R Icons. In Robert Biddle and Bruce Thomas, editors, *User Interfaces 2003: Fourth Australian User Interface Conference*, pages 25–32, 2003.

[10] Max Black. *Models and Metaphors.* Cornell University Press, 1962.

[11] Alan Blackwell. *Metaphor in Diagrams.* PhD thesis, University of Cambridge, September 1998.

[12] Lesley Brown, editor. *The New Shorter Oxford English Dictionary*, volume I (A-M). Oxford University Press, 1993.

[13] Lesley Brown, editor. *The New Shorter Oxford English Dictionary*, volume II (N-Z). Oxford University Press, 1993.

[14] John M. Carroll and Robert L. Mack. Learning to use a word processor: By doing, by thinking, and by knowing. In Ronald M. Baecker, Jonathan Grudin, William A. S. Buxtin, and Saul Greenberg, editors, *Readings in Human-Computer Interaction: Toward the Year 2000*, pages 698–717. Morgan Kaufmann Publishers, Inc., 1995.

[15] John M. Carroll, Robert L. Mack, and Wendy A. Kellogg. Interface metaphors and user interface design. In M. Helander, editor, *Handbook of Human-Computer Interaction*, pages 67–85. Elsevier Science Publishers, 1988.

[16] John M. Carroll and John C. Thomas. Metaphor and the cognitive representation of computing systems. *IEEE Transactions on Systems, Man. and Cybernetics*, 12(2):107–116, March/April 1982.

[17] Tiziana Cataci, Maria F. Costabile, and Maristella Matera. Which metaphor for which database? In M. A. R. Kirby, A. J. Dix, and J. E. Finlay, editors, *People and Computers X: Proceedings of HCI'95*. Cambridge University Press, 1995.

[18] Daniel Chandler. *Semiotics: The Basics*. Routledge, 2002.

[19] Larry L. Constantine and Lucy A. D. Lockwood. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Addison-Wesley, 1999.

[20] Alan Cooper and Paul Saffo. *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How To Restore The Sanity*. Sams, 1999.

[21] Richard Coyne. *Designing Information Technology in the Postmodern Age: From Method to Metaphor*. MIT Press, 1995.

[22] Donald Davidson. *Inquiries into Truth and Interpretation*, chapter 17. Oxford University Press, 2001.

[23] Ferdinand de Saussure. *Course in General Linguistics*. McGraw-Hill, 1966.

[24] Clarisse Sieckenius de Souza. The semiotic engineering of user interface languages. *International Journal of Man-Machine Studies*, 39(5):753–773, 1993.

[25] Michael Dertouzos. *The Unfinished Revolution*. HarperCollins Publishers, Inc., 2001.

[26] Martin Dodge and Rob Kitchin. *The Atlas of Cyberspace*. Addison-Wesley Publishing Company, 2002.

[27] Umberto Eco. *A Theory of Semiotics*. Indiana University Press, 1976.

[28] Umberto Eco. *The Role of the Reader*. Indiana University Press, 1979.

[29] Umberto Eco. *The Limits of Interpretation*. Indiana University Press, 1990.

[30] Thomas Erickson. The interaction design patterns homepage. `http://www.pliant.org/personal/Tom_Erickson/InteractionPatterns.html`.

[31] Thomas Erickson. Supporting interdisciplinary design: Towards pattern languages for workplaces. `http://www.pliant.org/personal/Tom_Erickson/Patterns.Chapter.html`.

[32] Thomas D. Erickson. Working with interface metaphors. In Brenda Laurel, editor, *The Art of Human-Computer Interface Design*, pages 65–73. Addison-Wesley Publishing Company, 1990.

[33] J. J. Gibson. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, 1986.

[34] Joseph Goguen. An introduction to algebraic semotics, with applications to user interface design. In Chrystopher Nehaniv, editor, *Computation for Metaphor, Analogy and Agents*, volume 1562 of *LNAI*, pages 242–291. Springer-Verlag, 1999.

[35] A. J. Greimas. *Structural Semantics*. University of Nebraska Press, 1983.

[36] F. Halasz and T. P. Moran. Analogy considered harmful. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 383–386, 1982.

[37] Ted Honderich, editor. *The Oxford Companion to Philosophy*. Oxford University Press, 1995.

[38] Jeff Johnson, Teresa L. Roberts, William Verplank, David C. Smith, Charles H. Irby, Marian Beard, and Keven Mackey. The Xerox Star: A retrospective. *IEEE Computer*, 22(9):11–29, September 1989.

[39] Steven Johnson. *Interface Culture: How New Technology Transforms the Way We Create and Communicate*. Harper SanFrancisco, 1997.

[40] Alan Kay. User interface: A personal view. In Brenda Laurel, editor, *The Art of Human-Computer Interface Design*, pages 191–207. Addison-Wesley Publishing Company, 1990.

[41] J. E. Kendall and K. E. Kendall. Metaphors and their meaning for information systems development. *European Journal of Information Systems*, 3(1):37–47, January 1994.

[42] George Lakoff. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. The University of Chicago Press, 1987.

[43] George Lakoff and Mark Johnson. *Metaphors We Live By*. The University of Chicago Press, 1980.

[44] Brenda Laurel. *Computers as Theatre*. Addison-Wesley Publishing Company, Inc., 1993.

[45] Jay Lundell and Steve Anderson. Designing a 'Front Panel' for Unix: The evolution of a metaphor. In Irvin R. Katz, Robert Mack, and Linn Marks, editors, *Proceedings of CHI'95 Conference on Human Factors in Computing Systems*, pages 573–580, New York, 1995. Addison-Wesley Publishing Co.

[46] Kim Halskov Madsen. A guide to metaphorical design. *Communications of the ACM*, 37(12):57–62, December 1994.

[47] Grant Malcolm and Joseph A. Goguen. Signs and representations: Semiotics for user interface design. In Ray Paton and Irene Nielson, editors, *Visual Representations and Interpretations*, pages 163–172. Springer, 1998.

[48] Microsoft Corporation. *The Windows Interface Guidelines for Software Design: An Application Design Guide*. Microsoft Press, 1995.

[49] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[50] Charles Morris. *Foundations of a Theory of Signs*. University of Chicago Press, 1938.

[51] S. Joy Mountford. Tools and techniques for creative design. In Brenda Laurel, editor, *The Art of Human-Computer Interface Design*, pages 17–30. Addison-Wesley Publishing Company, 1990.

[52] Kevin Mullet and Darrell Sano. *Designing Visual Interfaces: Communication Oriented Techniques*. SunSoft Press, 1995.

[53] Mihai Nadin. Interface design: A semiotic paradigm. *Semiotica*, 69(3):269–302, 1988.

[54] Theodor Holm Nelson. The right way to think about software design. In Brenda Laurel, editor, *The Art of Human-Computer Interface Design*, pages 235–243. Addison-Wesley Publishing Company, 1990.

[55] Jakob Nielsen. How to conduct a heuristic evaluation. `http://www.useit.com/papers/heuristic/heuristic_evaluation.html`.

[56] Jakob Nielsen. A meta-model for interacting with computers. *Interacting With Computers*, 2(2):147–160, 1990.

[57] Jakob Nielsen. *Usability Engineering*. Academic Press, 1993.

[58] Jakob Nielsen. Guerilla HCI: Using discount usability engineering to penetrate the intimidation barrier. In R. G. Bias and D. J. Mayhew, editors, *Cost-justifying usability*, pages 242–272. Academic Press, 1994.

[59] Jakob Nielsen. *Heuristic Evaluation*. John Wiley & Sons, 1994.

[60] Jakob Nielsen. *Designing Web Usability*. New Riders, 2000.

[61] Jakon Nielsen. Severity ratings for usability problems. `http://www.useit.com/papers/heuristic/severityrating.html`.

[62] James Noble and Robert Biddle. Patterns as signs. In Boris Magnusson, editor, *16th European Conference on Object-Oriented Programming*, pages 368–391, 2002.

[63] Donald A. Norman. *The Design of Everyday Things*. Doubleday, 1988.

[64] Donald A. Norman. *The Invisible Computer*. The MIT Press, 1998.

[65] Joanna Oja, Jere Mäkelä, and Mika Metsola. Programming assignment specification. `http://www.tml.hut.fi/Studies/T-110.350/2003/Program/specs2003.html`, 2003.

[66] Jean-Marc Orliaguet. How do we reason when using computers? how programmable are we? `http://www.medialab.chalmers.se/people/jmo/essays/how_do_we_reason.pdf%`, 2000.

[67] Charles Sanders Peirce. *Collected Papers*. four volumes. Harvard University Press, 1934–1948.

[68] Marcelo Soares Pimenta and Richard Faust. HCI and requirements engineering - eliciting interactive systems requirements in a language-centred user-designer collaboration: A semiotic approach. *SIGCHI Bulletin*, 29(1), January 1997.

[69] Steven Pinker. *How The Mind Works*. Penguin Books Australia, 1997.

[70] Byron Reeves and Clifford Nass. *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*. Cambridge University Press, 1996.

[71] I. A. Richards. *The Philosophy of Rhetoric*. Oxford University Press, 1936.

[72] Diego C. Ruspini, Krasimir Kolarov, and Oussama Khatib. The haptic display of complex graphical environments. In *Computer Graphics (SIGGRAPH 97 Conference Proceedings)*, pages 345–352. ACM SIGGRAPH, 1997.

[73] Ben Schneiderman. *Designing the User Interface : Strategies for Effective Human-Computer Interaction*. Addison-Wesley, 3rd edition, 1997.

[74] William Shakespeare. *Romeo and Juliet*. Morrison and Gibb, Ltd., 1933.

[75] Michael Smyth, Ben Anderson, and James L. Alty. Metaphor reflections and a tool for thought. In M. A. R. Kirby, A. J. Dix, and J. E. Finlay, editors, *People and Computers X: Proceedings of HCI'95*. Cambridge University Press, 1995.

[76] Gerard Steen. *Understanding Metaphor in Literature*. Longman Group Limited, 1994.

[77] Mark Stefik. *Internet Dreams*. The MIT Press, 1996.

[78] Rob Swigart. A writer's desktop. In Brenda Laurel, editor, *The Art of Human-Computer Interface Design*, pages 135–141. Addison-Wesley Publishing Company, 1990.

[79] Tony Thwaites, Lloyd Davis, and Warwick Mules. *Tools for cultural studies: an introduction*. Macmillan Education, 1994.

[80] Bruce Tognazzini. *Tog on Interface*. Addison Wesley, 1992.

[81] Claire Tristram. The next computer interface. *Technology Review*, pages 52–59, December 2001.

[82] Kaisa Väänänen and Jens Schmidt. User interface for hypermedia: How to find good metaphors? In Catherine Plaisant, editor, *Proceedings of the CHI'94 Conference Companion on Human Factors in Computing Systems*, pages 263–264, 1994.

[83] Martijn van Welie. Interaction design patterns. `http://www.welie.com/patterns/`.

[84] Cathleen Wharton, John Rieman, Clayton Lewis, and Peter Polson. The cognitive walkthrough method: A practitioner's guide. In *Usability Inspection Methods*, chapter 5. John Wiley & Sons, 1994.

[85] Lucy Anne Wozny. The application of metaphor, analogy, and conceptual models in computer systems. *Interacting with Computers*, 1(3):273–283, 1989.