Chapter 2: Understanding and Conceptualizing Interaction

2.1 Introduction

Imagine you have been asked to design an application to enable people to share their photos, movies, music, chats, documents, and so on in an efficient, safe, and enjoyable way. What would you do? How would you start? Would you begin by sketching out how the interface might look, work out how the system architecture should be structured, or simply start coding? Or, would you start by asking users about their current experiences of sharing files and look at existing tools, e.g., Dropbox, and, based on this, begin thinking about why and how you were going to design the application?

Interaction designers would begin by doing the latter. It is important to realize that having a clear understanding of why and how you are going to design something, before writing any code, can save enormous amounts of time, effort, and money later on in the design process. Ill thought-out ideas, incompatible and unusable designs can be refined while it is relatively easy and painless to do so. Once ideas are committed to code they become much harder to throw away. Such preliminary thinking through of ideas about the user experience and what kinds of designs might be appropriate is, however, a skill that needs to be learned. It is not something that can be done overnight by following a checklist, but requires practice in learning to identify, understand, and examine the issues—just like learning to write an essay or to program. In this chapter we describe the steps involved. In particular, we focus on what it takes to understand and conceptualize interaction.

2.2 Understanding the Problem Space and Conceptualizing Design

In the process of creating an interactive product, it can be tempting to begin at the nuts and bolts level of design. By this, we mean working out how to design the physical interface and what technologies and interaction styles to use, e.g., whether to use multitouch, speech, graphical user interface, head-up display, augmented reality, gesture-based, etc. The problem with starting here is that usability and user experience goals (which we describe in Chapter 1) can be overlooked. For example, consider the possibility of designing an integrated in-car entertainment, phone, and navigation system that allows drivers to follow directions, find nearby eating places, watch TV (already possible in Korea—see Figure 2.1a), and read their email. Such a gadget might seem attractive to some, offering drivers more choice: they can keep an eye on live sports games, find if there is a Cozy Coffee Shop in the next town, and so on. However, you might already be thinking ‘How distracting is that?’ Now imagine how new projection technology could be used as part of the system—instead of displaying the different kinds of information all on one small display that has to be toggled through, it could be displayed throughout the vehicle, on the dashboard, the rear-view mirror, and the windshield (see Figure 2.1b). However, this is likely to be even more dangerous—it could easily distract drivers, encouraging them to switch their attention from the road to the various images being projected.
While it is certainly necessary at some point to choose which technology to employ and decide how to design the physical aspects, it is better to make these kinds of decisions after articulating the nature of the problem space. By this we mean understanding and conceptualizing what is currently the user experience/product and how this is going to be improved or changed. This requires a design team thinking through how their ideas will support or extend the way people communicate and interact in their everyday activities. In the above example, it involves finding out what is problematic with existing forms of navigating while driving, e.g., trying to read maps while moving the steering wheel or looking at a small GPS display mounted on the dashboard when approaching a roundabout, and how to ensure that drivers can continue to drive safely without being distracted.

As emphasized in Chapter 1, identifying usability and user experience goals is a prerequisite to understanding the problem space. Another important consideration is to make explicit underlying assumptions and claims. By an assumption is meant taking something for granted when it needs further investigation, e.g., people will want to watch TV while driving. By a claim is meant stating something to be true when it is still open to question, e.g., a multimodal style of interaction for controlling a car navigation system – one that involves speaking while driving – is perfectly safe. Writing down your assumptions and claims and then trying to defend and support them can highlight those that are vague or wanting. In so doing, poorly constructed design ideas can be reformulated. In many projects, this process involves identifying human activities and interactivities that are problematic and working out how they might be improved through being supported with a different set of functions. In others, it can be more speculative, requiring thinking through what to design for an engaging user experience that does not exist.

The process of articulating the problem space is typically done as a team effort. Invariably, team members will have differing perspectives on the problem space. For example, a project manager is likely to be concerned about a proposed solution in terms of budgets, timelines, and staffing costs, whereas a software engineer will be thinking about breaking it down into specific technical concepts. It is important that the implications of pursuing each perspective are considered in relation to one another. Although time-consuming and sometimes resulting in disagreements among the team, the benefits of this process can far outweigh the associated costs: there will be much less chance of incorrect assumptions and unsupported claims creeping into a design solution that later turn out to be unusable or unwanted. Furthermore, spending time enumerating and reflecting upon ideas during the early stages of the design process enables more options and possibilities to be considered. Box 2.1 presents a hypothetical scenario of a team working through their assumptions and claims, showing how, in so doing, problems are explicated and explored, leading to a specific avenue of investigation agreed on by the team.

**BOX 2.1: A hypothetical scenario of early design highlighting the assumptions and claims (italicized) made by different members of a design team**

A large software company has decided it needs to develop an upgrade of its web browser for smartphones because its marketing team has discovered that many of the company's customers have switched over to using another mobile browser. The marketing people assume something is wrong with their browser and that their rivals have a better product. But they don't know what the problem is with theirs. The design team put in charge of this project assume they need to improve the usability of a number of the browser's functions. They claim that this will win back users by making features of the interface simpler, more attractive, and more flexible to use.

The user experience researchers on the design team conduct an initial user study investigating how people use the company's web browser on a variety of smartphones. They also look at other mobile web browsers on the market and compare their functionality and usability. They observe and talk to many different users. They discover several things about the usability of their web browser, some of which they were not expecting. One revelation is that many of their customers have never actually used the bookmarking tool. They present their findings to the rest of the team and have a long discussion about why each of them thinks it is not being used. One member claims that the web browser's function...
for organizing bookmarks is fiddly and error-prone – especially when using a multitouch screen – and assumes this is the reason why many users do not use it. Another member backs her up, saying how awkward it is to use this method when wanting to move bookmarks between folders. One of the user experience architects agrees, noting how several of the users he talked to mentioned how difficult and time-consuming they found it when trying to move bookmarks between folders and how they often ended up accidentally putting them into the wrong folders.

A software engineer reflects on what has been said, and makes the claim that the bookmark function is no longer needed since he assumes that most people do what he does, which is to revisit a website by flicking through their history list of previously visited pages. Another member of the team disagrees with him, claiming that many users do not like to leave a trail of the sites they have visited and would prefer to be able to save only sites they think they might want to revisit. The bookmark function provides them with this option. Another option discussed is whether to include most frequently visited sites as thumbnail images or as tabs. The software engineer agrees that providing all options could be a solution but worries how this might clutter the small screen interface.

After much discussion on the pros and cons of bookmarking versus history lists, the team decides to investigate further how to support effectively the saving, ordering, and retrieving of websites using a mobile web browser. All agree that the format of the existing web browser’s structure is too rigid and that one of their priorities is to see how they can create a simpler way of revisiting websites on the smartphone.

Explicating people’s assumptions and claims about why they think something might be a good idea (or not) enables the design team as a whole to view multiple perspectives on the problem space and, in so doing, reveal conflicting and problematic ones. The following framework is intended to provide a set of core questions to aid design teams in this process:

- Are there problems with an existing product or user experience? If so, what are they?
- Why do you think there are problems?
- How do you think your proposed design ideas might overcome these?
- If you have not identified any problems and instead are designing for a new user experience, how do you think your proposed design ideas support, change, or extend current ways of doing things?

ACTIVITY 2.1

Use the framework in the above list to explicate the main assumptions and claims behind 3D TV.

Comment

3D TV first went on sale in 2010. There was much hype and fanfare about the enhanced user experience it would offer, especially when watching movies, sports events, and dramas. An assumption was that people would not mind wearing the glasses that are needed to see in 3D, nor would they mind paying a lot more for a new 3D-enabled TV screen. A claim was that people would really enjoy the enhanced clarity and color detail provided by 3D, based on the favorable feedback received worldwide when viewing 3D films, such as Avatar, at a cinema.

Figure 2.2: A family enjoying the 3D TV experience

But the unanswered question was: Could the enhanced cinema experience become a desired living room experience? There is no existing problem to overcome – what is being proposed is a new way of experiencing TV. A user experience question is: How will people take to wearing the special glasses in their living room? Will it become the norm for people to
Having a good understanding of the problem space greatly helps design teams to then be able to conceptualize the design space. Primarily this involves articulating the proposed system and the user experience. The benefits of conceptualizing the design space early on are:

- **Orientation** – enabling the design team to ask specific kinds of questions about how the conceptual model will be understood by the targeted users.
- **Open-mindedness** – preventing the design team from becoming narrowly focused early on.
- **Common ground** – allowing the design team to establish a set of common terms that all can understand and agree upon, reducing the chance of misunderstandings and confusion arising later on.

Once formulated and agreed upon, a conceptual model can then become a shared blueprint. This can be represented as a textual description and/or in a diagrammatic form, depending on the preferred lingua franca used by the design team. As you will see later in Chapter 11, the conceptual model is used by the design team as the basis from which to develop more detailed and concrete aspects of the design.

### 2.3 Conceptual Models

How do you develop a conceptual model and how do you know you have a good one? We begin to address these questions here by drawing on Johnson and Henderson's (2002) account of a conceptual model. They describe one as "a high-level description of how a system is organized and operates" (Johnson and Henderson, 2002, p. 26). In this sense, it is an abstraction outlining what people can do with a product and what concepts are needed to understand how to interact with it. A key benefit of conceptualizing a design at this level is that it enables "designers to straighten out their thinking before they start laying out their widgets" (Johnson and Henderson, 2002, p. 28).

In a nutshell, a conceptual model provides a working strategy and a framework of general concepts and their interrelations. The core components are:

- Metaphors and analogies that convey to people how to understand what a product is for and how to use it for an activity (e.g. browsing, bookmarking).
- The concepts that people are exposed to through the product, including the task-domain objects they create and manipulate, their attributes, and the operations that can be performed on them (e.g. saving, revisiting, organizing).
- The relationships between those concepts (e.g. whether one object contains another, the relative importance of actions to others, and whether an object is part of another).
- The mappings between the concepts and the user experience the product is designed to support or invoke (e.g. one can revisit through looking at a list of visited sites, most frequently visited, or saved websites).

How the various metaphors, concepts, and their relationships are organized determines the user experience. By explicating these, the design team can debate the merits of providing different methods and how they support the main concepts, e.g. saving, revisiting, categorizing, reorganizing, and their mapping to the task domain. They can also begin discussing whether a new overall metaphor may be preferable that combines the activities of browsing, searching, and revisiting. In turn, this can lead the design team to articulate the kinds of relationships between them, such as containership. For example, what is the best way to sort and revisit saved pages and how many and what types of containers should be used (e.g. folders, bars, panes)? The same enumeration of concepts can be repeated for other functions of the web browser – both current and new. In so doing, the design team can begin to systematically work out what will be the most simple, effective, and memorable way of supporting users while browsing the Internet.

The best conceptual models are those that appear obvious; the operations they support being intuitive to use. However, sometimes applications can end up based on overly complex conceptual models, especially if they are the result of a series of upgrades, where more and more functions and ways of doing something are added to the original conceptual model. Whereas in the first version of the software there may have been one way of doing something, later versions are often designed to allow several ways of performing the same operation. For example, operating systems and word processors now make it possible for the user to carry out the same activity in a number of different ways, e.g. to delete a file the user can issue a command like Ctrl+D, speak to the computer by saying ‘delete file,’ or drag an icon of the file to the recycle bin. Users have to learn each of the different styles to decide which they prefer. Many users prefer to stick to
the methods they have always used and trusted and, not surprisingly, become annoyed when they find a simple way of doing something has been changed, albeit more flexibly, now allowing them to do it in three or more different ways. The benefits of providing multiple ways of carrying out the same operation need to be weighed against a constrained interface that offers only one way of performing an operation.

Most interface applications are actually based on well-established conceptual models. For example, a conceptual model based on the core aspects of the customer experience when at a shopping mall underlies most online shopping websites. These include the placement of items a customer wishes to purchase into a shopping cart or basket and proceeding to checkout when ready to make the purchase. A variation – which is also based on what happens in a physical store – is making a booking, where new items are added, before proceeding to pay. Collections of patterns are now readily available to help design the interface for these core transactional processes – together with many other aspects of a user experience – meaning interaction designers do not have to start from scratch every time they design or redesign an application. Examples include patterns for online forms and event calendars (for more on these, see Chapter 11).

**BOX 2.2: Design concept**

Another term that is sometimes used is a design concept – essentially it is a set of ideas for a design. Typically, it comprises scenarios, images, mood boards, or text-based documents. For example, Figure 2.3 shows the first page of a design concept developed for an ambient display that was aimed at changing people’s behavior in a building. Part of the design concept was envisioned as an animated pattern of twinkly lights that would be embedded in the carpet near the entrance of the building with the intention of luring people towards the stairs (Hazlewood et al., 2010).

![Figure 2.3: The first page of a design concept for an ambient display](image)

Hence, it is rare for completely new conceptual models to emerge that transform the way we carry out our everyday and work activities when using a computer. The classics include the desktop (developed by Xerox in the late 1970s), the spreadsheet (developed by Bricklin and Frankston in the late 1970s), and the web (developed by Berners Lee in the early 1980s). All have made what was previously limited to a few skilled people accessible to all, while greatly expanding what is possible. The first dramatically changed how office tasks can be performed (including creating, editing, and printing documents); the second made accounting highly flexible and easier to accomplish, enabling a diversity of new computations to be performed simply through filling in interactive boxes; and the third allowed anyone to remotely browse a network of information. More recently, e-readers, such as the Kindle, have introduced a new way of reading a book while multitouch interfaces, such as the iPad, have introduced a novel way of interacting with digital information. Importantly, all of these conceptual models were based on familiar activities that have greatly transformed them.

**BOX 2.3: A classic conceptual model: the Star**

The Star interface developed by Xerox back in 1981 (see Figure 2.4) revolutionized the way interfaces were designed for personal computing (Smith et al., 1982; Miller and Johnson, 1996). It was designed as an office system, targeted at workers not interested in computing per se and was based on a conceptual model that included the familiar knowledge of an office. Paper, folders, filing cabinets, and mailboxes were represented as icons on the screen and were designed to possess some of the properties of their physical counterparts. Dragging a document icon across the desktop screen was
seen as equivalent to picking up a piece of paper in the physical world and moving it (but this, of course, is a very different action). Similarly, dragging an electronic document onto an electronic folder was seen as being analogous to placing a physical document into a physical cabinet. In addition, new concepts that were incorporated as part of the desktop metaphor were operations that could not be performed in the physical world. For example, electronic files could be placed onto an icon of a printer on the desktop, resulting in the computer printing them out.

![Image](image.png)

**Figure 2.4:** (a) The Xerox Star and (b) GUI interface

### 2.4 Interface Metaphors

As mentioned earlier, metaphors are considered to be a central component of a conceptual model. They provide a structure that is similar in some way to aspects of a familiar entity (or entities) but also have their own behaviors and properties. More specifically, an interface metaphor is one that is instantiated in some way as part of the user interface: for example, the desktop metaphor. Another well known one is the search engine. This term was originally coined in the early 1990s to refer to a software tool that indexed and retrieved files remotely from the Internet, using various algorithms to match terms selected by the user. The metaphor invites comparisons between a mechanical engine, which has several parts working, and the everyday action of looking in different places to find something. The functions supported by a search engine also include other features besides those belonging to an engine that searches, such as listing and prioritizing the results of a search. It also does these actions in quite different ways from how a mechanical engine works or how a human being might search a library for books on a given topic. The similarities implied by the use of the term search engine, therefore, are at a general level. They are meant to conjure up the essence of the process of finding relevant information, enabling the user to link these to less familiar aspects of the functionality provided.

**ACTIVITY 2.2**

Go to a few online stores and see how the interface has been designed to enable the customer to order and pay for an item. How many use the ‘add to shopping cart/trolley/basket’ followed by the ‘checkout’ metaphor? Does this make it straightforward and intuitive to make a purchase?

**Comment**

Making a purchase online is an undertaking with risks and people want to feel they are making the right choice. Designing the interface to have a familiar metaphor (with an icon of a shopping cart/basket – although not a cash till!) makes it easier for people to know what to do at the different stages of making a purchase. Importantly, placing an item in the basket does not commit the customer to purchase it there and then. It also enables them to browse further and select other items – as they might in a physical store.

Interface metaphors are intended to provide familiar entities that enable people to readily understand the underlying conceptual model and know what to do at an interface. However, they can also contravene people’s expectations about how things should be, such as the recycle bin (trashcan) that used to sit on the desktop. Logically and culturally (i.e. in the real world) it should have been placed under the desk. But users would not have been able to see it because it would be occluded by the desktop surface. So it needed to go on the desktop. Some users find this irksome but most did not find it to be a problem. Once they understood why the bin icon was on the desktop they simply accepted it being there.

Other times, designers can fall into the trap of trying to create a virtual object to resemble a familiar physical object that is itself badly designed. A well-known example is the virtual calculator, which is designed to look and behave like a physical calculator. The interface of some physical calculators, however, have been poorly designed in the first place, based on
poor conceptual models, with excessive use of modes, poor labeling of functions, and difficult-to-manipulate key sequences (Mullet and Sano, 1995). The design of the calculator in Figure 2.5(a) has even gone as far as replicating functions needing shift keys (e.g. deg, oct, and hex), which could have been redesigned as dedicated software buttons. Trying to use a virtual calculator that has been designed to emulate a poorly designed physical calculator is much harder than using the physical device itself. A better approach is to think about the kinds of calculations people typically want to do when using their phones or computers. For example, the Mac calculator in Figure 2.5(b) has a much simpler interface that supports a basic set of calculations, which is all that most people need to do (although how many people use the top four memory keys?).

Figure 2.5: Two digital calculators where (a) has been designed too literally and (b) more appropriately for a computer screen

In many cases, new interface metaphors rapidly become integrated into common parlance, as witnessed by the way people talk about them. People surf the net, poke their friends, and leave messages on their wall in the same way they would talk about winning an argument or saving time. As such, the interface metaphors are no longer talked about as familiar terms to describe less familiar computer-based actions; they have become everyday terms in their own right. Moreover, it is hard not to use metaphorical terms when talking about technology use, as they have become so ingrained in the language we use to express ourselves. Just ask yourself or someone else to describe how the Internet works. Then try doing it without using a single metaphor.

**BOX 2.4: Why are metaphors so popular?**

People frequently use metaphors and analogies (here we use the terms interchangeably) as a source of inspiration for understanding and explaining to others what they are doing, or trying to do, in terms that are familiar to them. They are an integral part of human language (Lakoff and Johnson, 1980). Metaphors are commonly used to explain something that is unfamiliar or hard to grasp by way of comparison with something that is familiar and easy to grasp. For example, they are commonly employed in education, where teachers use them to introduce something new to students by comparing the new material with something they already understand. An example is the comparison of human evolution with a game. We are all familiar with the properties of a game: there are rules, each player has a goal to win (or lose), there are heuristics to deal with situations where there are no rules, there is the propensity to cheat when the other players are not looking, and so on. By conjuring up these properties, the analogy helps us begin to understand the more difficult concept of evolution – how it happens, what rules govern it, who cheats, and so on.

It is not surprising, therefore, to see how widely metaphors have been used in interaction design to conceptualize abstract, hard to imagine, and difficult to articulate computer-based concepts and interactions in more concrete and familiar terms and as graphical visualizations at the interface. Metaphors and analogies are used in three main ways:

1. As a way of conceptualizing what we are doing (e.g. surfing the web).
2. As a conceptual model instantiated at the interface (e.g. the desktop metaphor).
3. As a way of visualizing an operation (e.g. an icon of a shopping cart into which we place items we wish to purchase on an online shopping site).
2.5 Interaction Types

Another way of conceptualizing the design space is in terms of the interaction types that will underlie the user experience. Essentially, these are the ways a person interacts with a product or application. We propose that there are four main types: instructing, conversing, manipulating, and exploring. Deciding upon which of these to use, and why, can help designers formulate a conceptual model before committing to a particular interface in which to implement them, e.g. speech-based, gesture-based, touch-based, menu-based, and so on. Note that we are distinguishing here between interaction types (which we discuss in this section) and interface types (which will be discussed in Chapter 6). While cost and other product constraints will often dictate which interface style can be used for a given application, considering the interaction type that will best support a user experience can highlight the potential trade-offs, dilemmas, and pros and cons.

Consider the following problem description: a company has been asked to design a computer-based system that will encourage autistic children to communicate and express themselves better. What type of interaction would be appropriate to use at the interface for this particular user group? It is known that autistic children find it difficult to express what they are feeling or thinking through talking and are more expressive when using their bodies and limbs. Clearly an interaction style based on talking would not be effective but one that involves the children interacting with a system by moving in a physical and/or digital space would seem a more promising starting point.

Below we describe in more detail each of the four types of interaction. It should be noted that they are not meant to be mutually exclusive (e.g. someone can interact with a system based on different kinds of activities); nor are they meant to be definitive.

1. Instructing – where users issue instructions to a system. This can be done in a number of ways, including: typing in commands, selecting options from menus in a windows environment or on a multitouch screen, speaking aloud commands, gesturing, pressing buttons, or using a combination of function keys.
2. Conversing – where users have a dialog with a system. Users can speak via an interface or type in questions to which the system replies via text or speech output.
3. Manipulating – where users interact with objects in a virtual or physical space by manipulating them (e.g. opening, holding, closing, placing). Users can hone their familiar knowledge of how to interact with objects.
4. Exploring – where users move through a virtual environment or a physical space. Virtual environments include 3D worlds, and augmented and virtual reality systems. They enable users to hone their familiar knowledge of physically moving around. Physical spaces that use sensor-based technologies include smart rooms and ambient environments, also enabling people to capitalize on familiarity.

Besides these core activities of instructing, conversing, manipulating, and exploring, it is possible to describe the specific domain and context-based activities users engage in, such as learning, working, socializing, playing, browsing, writing, problem-solving, decision-making, and information-searching – to name but a few. McCullough (2004) suggests describing them as situated activities, organized by: work (e.g. presenting to groups), home (e.g. resting), in town (e.g. eating), and on the road (e.g. walking). The rationale is to help designers be less ad hoc and more systematic when thinking about the usability of technology-modified places in the environment. Below we illustrate in more detail our four core interaction types and how to design applications for them.

2.5.1 Instructing

This type of interaction describes how users carry out their tasks by telling the system what to do. Examples include giving instructions to a system to perform operations such as tell the time, print a file, and remind the user of an appointment. A diverse range of products has been designed based on this model, including home entertainment systems, consumer electronics, and computers. The way in which the user issues instructions can vary from pressing buttons to typing in strings of characters. Many activities are readily supported by giving instructions.

Operating systems like Unix and Linux have been designed primarily as command-based systems, where users issue instructions at the prompt as a command or set of commands. In Windows and other GUI-based systems, control keys or the selection of menu options via a mouse, touch pad, or touch screen are used. Typically, a wide range of functions are provided from which users have to select when they want to do something to the object on which they are working. For example, a user writing a report using a word processor will want to format the document, count the number of words typed, and check the spelling. The user instructs the system to do these operations by issuing appropriate commands. Typically, commands are carried out in a sequence, with the system responding appropriately (or not) as instructed.

One of the main benefits of designing an interaction based on issuing instructions is that the interaction is quick and efficient. It is particularly fitting where there is a need to frequently repeat actions performed on multiple objects. Examples include the repetitive actions of saving, deleting, and organizing files.
There are many different kinds of vending machines in the world. Each offers a range of goods, requiring the user initially to part with some money. Figure 2.6 shows photos of two different vending machines, one that provides soft drinks and the other a range of snacks. Both use an instructional mode of interaction. However, the way they do so is quite different.

What instructions must be issued to obtain a soda from the first machine and a bar of chocolate from the second? Why has it been necessary to design a more complex mode of interaction for the second vending machine? What problems can arise with this mode of interaction?

Comment

The first vending machine has been designed using simple instructions. There are a small number of drinks to choose from and each is represented by a large button displaying the label of each drink. The user simply has to press one button and this should have the effect of returning the selected drink. The second machine is more complex, offering a wider range of snacks. The trade-off for providing more options, however, is that the user can no longer instruct the machine by using a simple one-press action but is required to use a more complex process, involving (i) reading off the code (e.g. C12) under the item chosen, then (ii) keying this into the number pad adjacent to the displayed items, and (iii) checking the price of the selected option and ensuring that the amount of money inserted is the same or greater (depending on whether or not the machine provides change). Problems that can arise from this type of interaction are the customer misreading the code and/or miskeying the code, resulting in the machine not issuing the snack or providing the wrong item.

A better way of designing an interface for a large number of options of variable cost might be to continue to use direct mapping, but use buttons that show miniature versions of the snacks placed in a large matrix (rather than showing actual versions). This would use the available space at the front of the vending machine more economically. The customer would need only to press the button of the object chosen and put in the correct amount of money. There is less chance of error resulting from pressing the wrong code or keys. The trade-off for the vending company, however, is that the machine is less flexible in terms of which snacks it can sell. If a new product line comes out they will also need to replace part of the physical interface to the machine – which would be costly.

2.5.2 Conversing

This form of interaction is based on the idea of a person having a conversation with a system, where the system acts as a dialog partner. In particular, the system is designed to respond in a way another human being might when having a conversation. It differs from the activity of instructing insofar as it encompasses a two-way communication process with the system acting like a partner rather than a machine that obeys orders. It has been most commonly used for applications where the user needs to find out specific kinds of information or wants to discuss issues. Examples include advisory systems, help facilities, and search engines.
The kinds of conversation that are currently supported range from simple voice-recognition, menu-driven systems that are interacted with via phones, to more complex natural language-based systems that involve the system parsing and responding to queries typed in by the user. Examples of the former include banking, ticket booking, and train-time inquiries, where the user talks to the system in single-word phrases and numbers – e.g. yes, no, three – in response to prompts from the system. Examples of the latter include search engines and help systems, where the user types in a specific query – e.g. ‘how do I change the margin widths?’ – to which the system responds by giving various answers.

A main benefit of developing a conceptual model that uses a conversational style of interaction is that it allows people, especially novices, to interact with a system in a way that is familiar to them. For example, the search engine Ask Jeeves for Kids! allows children to ask a question in a way they would when asking their teachers or parents rather than making them reformulate their question in terms of keywords and Boolean logic. Similarly, the generation of virtual representatives that have been incorporated into online store websites offer customers quick and direct answers to their product-related queries. An example is Anna, whose appearance was commented upon in the last chapter. She is a semi-cartoon character fronting the Swedish furniture store Ikea's Help Center (www.ikea.com) by directing the user to a part of the store's website in response to his questions typed in at the dialog box (see Figure 2.7). For example, when a user types in ‘do you have any kitchen chairs?’ Anna replies ‘please have a look at the chairs’ and a page of chairs is automatically displayed. The system matches keywords in the queries to a database of suitable web pages or answers.

A problem that can arise from using a conversational-based interaction type is that certain kinds of tasks are transformed into cumbersome and one-sided interactions. This is especially true for automated phone-based systems that use auditory menus to advance the interaction. Users have to listen to a voice providing several options, then make a selection, and repeat through further layers of menus before accomplishing their goal, e.g. reaching a real human or paying a bill. Here is the beginning of a dialog between a user who wants to find out about car insurance and an insurance company’s reception system:

<user dials an insurance company>

‘Welcome to St. Paul's Insurance Company. Press 1 if you are a new customer; 2 if you are an existing customer.’

<user presses 1>

‘Thank you for calling St. Paul's Insurance Company. If you require house insurance press 1, car insurance press 2, travel insurance press 3, health insurance press 4, other press 5.’

<user presses 2>

‘You have reached the car insurance division. If you require information about fully comprehensive insurance press 1, third-party insurance press 2 …’
2.5.3 Manipulating

This form of interaction involves manipulating objects and capitalizes on users’ knowledge of how they do so in the physical world. For example, digital objects can be manipulated by moving, selecting, opening, and closing. Extensions to these actions include zooming in and out, stretching, and shrinking – actions that are not possible with objects in the real world. Human actions can be imitated through the use of physical controllers (e.g. Wii) or gestures made in the air (e.g. Kinect) to control the movements of an on-screen avatar. Physical toys and robots have also been embedded with computation and capability that enables them to act and react in programmable ways depending on whether they are squeezed, touched, sensed, or moved. Tagged physical objects (e.g. balls, bricks, blocks) that are manipulated in a physical world (e.g. placed on a surface) can result in other physical and digital events occurring, such as a lever moving or a sound or animation being played.

A framework that has been highly influential in informing the design of GUI applications is direct manipulation (Shneiderman, 1983). It proposes that digital objects be designed at the interface so that they can be interacted with in ways that are analogous to how physical objects in the physical world are manipulated. In so doing, direct manipulation interfaces are assumed to enable users to feel that they are directly controlling the digital objects represented by the computer. The three core principles are:

1. continuous representation of the objects and actions of interest;
2. rapid reversible incremental actions with immediate feedback about the object of interest;
3. physical actions and button pressing instead of issuing commands with complex syntax.

According to these principles, an object on the screen remains visible while a user performs physical actions on it and any actions performed on it are immediately visible. For example, a user can move a file by dragging an icon that represents it from one part of the desktop to another. The benefits of direct manipulation include:

- helping beginners learn basic functionality rapidly;
- enabling experienced users to work rapidly on a wide range of tasks;
- allowing infrequent users to remember how to carry out operations over time;
- preventing the need for error messages, except very rarely;
- showing users immediately how their actions are furthering their goals;
- reducing users’ experiences of anxiety;
- helping users gain confidence and mastery and feel in control.

Apple was one of the first computer companies to design an operating environment that used direct manipulation as its central mode of interaction. The highly successful Mac desktops (Figures 2.8a and b) and the more recent iPad display (Figure 2.8c) show the evolution of direct manipulation interfaces over the past 25 years.
Many applications have been developed based on some form of direct manipulation, including word processors, video games, learning tools, and image editing tools. However, while direct manipulation interfaces provide a very versatile mode of interaction they do have their drawbacks. In particular, not all tasks can be described by objects and not all actions can be undertaken directly. Some tasks are also better achieved through issuing commands. For example, consider how you edit an essay using a word processor. Suppose you had referenced work by Ben Shneiderman but had spelled his name as ‘Schneiderman’ throughout the essay. How would you correct this error using a direct manipulation interface? You would need to read through your essay and manually select the ‘c’ in every ‘Schneiderman,’ highlighting and then deleting it. This would be very tedious and it would be easy to miss one or two. By contrast, this operation is relatively effortless and also likely to be more accurate when using a command-based interaction. All you need to do is instruct the word processor to find every ‘Schneiderman’ and replace it with ‘Shneiderman.’ This can be done through selecting a menu option or using a combination of command keys and then typing the changes required into the dialog box that pops up.

2.5.4 Exploring

This mode of interaction involves users moving through virtual or physical environments. For example, users can explore aspects of a virtual 3D environment, such as the interior of a building. Physical environments can also be embedded with sensing technologies that, when they detect the presence of someone or certain body movements, respond by triggering certain digital or physical events. Similar to direct manipulation and direct manipulatives, the fundamental idea is to enable people to explore and interact with an environment, be it physical or digital, by exploiting their knowledge of how they move and navigate through existing spaces.

Many 3D virtual environments have been built that include virtual worlds designed for people to move between various spaces to learn (e.g. virtual universities) and fantasy worlds where people wander around different places to socialize (e.g. virtual parties). One of the best-known examples is Second Life. Numerous virtual landscapes depicting cities, parks, buildings, rooms, and datasets have also been built, both realistic and abstract, that enable users to fly over them and zoom in and out of different parts. Other virtual environments that have been built include worlds that are larger than life, enabling users to move around them, experiencing things that are normally impossible or invisible to the eye (Figure 2.9a); highly realistic representations of architectural designs, allowing clients and customers to imagine how they will use and move through planned buildings and public spaces; and visualizations of complex datasets that scientists can virtually climb inside and experience (Figure 2.9b).
A number of physical environments have been developed using embedded sensor technologies and other location-detection technologies. They are often called context-aware environments: the location and/or presence of people in the vicinity of a sensing device is detected and based on this, the environment decides which digital information to provide on a device (e.g. a nearby coffee bar where friends are meeting) or which action to perform (e.g. changing lights in a room) that is considered relevant or useful to the person at a particular time and place. Many location-based virtual guides have been developed for cell phones, which provide relevant information about restaurants, historical buildings, and other places of interest as the person wanders near them. Physically embedded environments have also been designed to extend how children learn. For example, the Ambient Wood project was designed as an outdoor learning experience where a physical woodland was wired to present various forms of digital information to children, as they moved around it (Rogers et al., 2005). Depending on which part of the woodland they passed by (e.g. a particular kind of tree, a bush, a hole), an image would occasionally pop up on a screen they were carrying, or a sound was played via hidden speakers or heard through a special handheld audio device – the ambient horn (see Figure 2.10). The idea was to provide contextually relevant digital information that would enhance the usual physical experience available to children when exploring an outdoor world.

2.6 Paradigms, Theories, Models, and Frameworks

Other sources of inspiration and knowledge that are used to inform design and guide research are paradigms, theories, models, and frameworks (Carroll, 2003). These vary in terms of their scale and specificity to a particular problem space. A paradigm refers to a general approach that has been adopted by a community of researchers and designers for carrying out their work, in terms of shared assumptions, concepts, values, and practices. A theory is a well-substantiated explanation of some aspect of a phenomenon, for example, the theory of information processing that explains how the mind, or some aspect of it, is assumed to work. A model is a simplification of some aspect of human–computer interaction intended to make it easier for designers to predict and evaluate alternative designs. A framework is a set of interrelated concepts and/or a set of specific questions that is intended to inform a particular domain area (e.g. collaborative learning), online communities, or an analytic method (e.g. ethnographic studies).

2.6.1 Paradigms

To follow a particular paradigm means adopting a set of practices that a community has agreed upon. These include:

- the questions to be asked and how they should be framed;
- the phenomena to be observed;
- the way in which findings from studies are to be analyzed and interpreted (Kuhn, 1972).

In the 1980s, the prevailing paradigm in human–computer interaction was how to design user-centered applications for the desktop computer. Questions about what and how to design were framed in terms of specifying the requirements for a single user interacting with a screen-based interface. Task analytic and usability methods were developed based on an
individual user's cognitive capabilities. The acronym WIMP was used as a way of characterizing the core features of an interface for a single user: this stood for Windows, Icons, Menus, and Pointer. This was later superseded by the GUI (graphical user interface), a term that has stuck with us ever since.

Within interaction design, many changes took place in the mid to late 1990s. The WIMP interface with its single thread, discrete event dialog was considered to be unnecessarily limiting (e.g. Jacob, 1996). Instead, many argued that a new paradigm was needed to enable more flexible forms of interaction to take place, having a higher degree of interactivity and parallel input/output exchanges. A shift in thinking, together with several technological advances, paved the way for a new method of conceptualizing human–computer interaction. The rhetoric ‘beyond the desktop’ became a pervasive starting point, resulting in many new challenges, questions, and phenomena being considered. New methods of designing, modeling, and analyzing came to the fore. At the same time, new theories, concepts, and ideas entered the stage. Turns to the social, the emotional, the environmental, and the wild began shaping what was studied, how it was studied, and ultimately what was designed. Significantly, one of the main frames of reference – the single user – was replaced by context.

A big influence in the more recent paradigmatic changes was Weiser’s (1991) vision of ubiquitous technology. He proposed that computers would become part of the environment, embedded in a variety of everyday objects, devices, and displays. He envisioned a world of serenity, comfort, and awareness, where people are kept perpetually informed of what is happening around them, what is going to happen, and what has just happened. Ubiquitous computing devices would enter a person's center of attention when needed and move to the periphery of their attention when not, enabling the person to switch calmly and effortlessly between activities without having to figure out how to use a computer when performing their tasks. In essence, the technology would be unobtrusive and largely disappear into the background. People would be able to get on with their everyday and working lives, interacting with information and communicating and collaborating with others without being distracted or becoming frustrated with technology.

Since the late 1990s many researchers have been concerned with how to embed and augment the environment with various computational resources to provide information and services, when and where desired. An assortment of sensors have been experimented with in our homes, hospitals, public buildings, physical environments, and even our bodies to detect trends and anomalies, providing a huge array of data about our health and movements, and changes in the environment. Algorithms have been developed to analyze the data in order for inferences to be drawn about what actions to take for people. In addition, sensed data are increasingly being used to automate mundane operations and actions that we would have done in our everyday worlds using conventional knobs, buttons, and other physical controls.

Many new challenges, themes, and questions have been articulated in interaction design and computer science (e.g. Rogers, 2006; Harper et al, 2008). These include:

- How to enable people to access and interact with information in their work, social, and everyday lives, using an assortment of technologies.
- How to design user experiences for people using interfaces that are part of the environment but where there are no obvious controlling devices.
- How and in what form to provide contextually relevant information to people at appropriate times and places to support them while on the move.
- How to ensure that information that is passed around via interconnected displays, devices, and objects is secure and trustworthy.

### 2.6.2 Theories

Over the past 30 years, numerous theories have been imported into human–computer interaction, providing a means of analyzing and predicting the performance of users carrying out tasks for specific kinds of computer interfaces and systems. These have been primarily cognitive, social, and organizational in origin. For example, cognitive theories about human memory were used in the 1980s to determine the best ways of representing operations, given people's memory limitations. One of the main benefits of applying such theories in interaction design is to help identify factors (cognitive, social, and affective) relevant to the design and evaluation of interactive products. Some of the most influential theories in HCI, including distributed cognition, will be covered in the next chapter.

### 2.6.3 Models

Models are typically abstracted from a theory coming from a contributing discipline, like psychology that can be directly applied to interaction design. For example, Norman (1988) developed a number of models of user interaction based on theories of cognitive processing, arising out of cognitive science, that were intended to explain the way users interacted with interactive technologies. These include the seven stages of action model that describes how users move from their plans to executing physical actions they need to perform to achieve them, to evaluating the outcome of their actions with
respect to their goals. Another highly influential model based on cognitive theory that made its mark in the 1980s was Card, Moran, and Newell's keystroke model (see Chapter 15). This was used by a number of researchers and designers as a predictive way of analyzing user performance for different interfaces to determine which would be the most effective. More recent models developed in interaction design are user models, which predict what information users want in their interactions, and models that characterize core components of the user experience, such as Norman's (2004) model of emotional design (Chapter 5).

2.6.4 Frameworks

Numerous frameworks have been introduced in interaction design to help designers constrain and scope the user experience for which they are designing. In contrast to a model – which is a simplification of a phenomenon – a framework offers advice to designers as to what to design or look for. This can come in a variety of forms, including steps, questions, concepts, challenges, principles, tactics, and dimensions. Frameworks, like models, have traditionally been based on theories of human behavior, but they are increasingly being developed from the experiences of actual design practice and the findings arising from user studies.

Many frameworks have been published in the HCI/interaction design literatures, covering different aspects of the user experience and a diversity of application areas. For example, there are frameworks for helping designers think about how to conceptualize learning, working, socializing, fun, emotion, and so on and others that focus on how to design particular kinds of technologies to evoke certain responses, e.g. persuasive technologies and pleasurable products (see Chapter 5). There are also general frameworks that show differences in perspective. For example, Table 2.1 shows part of a new framework proposed for human–computer interaction, contrasting old and new concepts (Rogers, 2009).

Table 2.1: A new framework for human–computer interaction (Rogers, 2009)

<table>
<thead>
<tr>
<th>Concern</th>
<th>Past</th>
<th>Future</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame of reference</td>
<td>• users</td>
<td>• context</td>
</tr>
<tr>
<td>Method, theory, and perspective</td>
<td>• scientific approach</td>
<td>• pluralistic</td>
</tr>
<tr>
<td></td>
<td>• interaction design</td>
<td>• mixing</td>
</tr>
<tr>
<td>Outputs</td>
<td>• ethnographies</td>
<td>• insights</td>
</tr>
<tr>
<td></td>
<td>• models and tools for analysis</td>
<td>• creating new ways of experiencing</td>
</tr>
<tr>
<td></td>
<td>• design guidance</td>
<td>• value-based analyses</td>
</tr>
</tbody>
</table>

A classic early example of a conceptual framework that has been highly influential in HCI is Norman’s (1988) explication of the relationship between the design of a conceptual model and a user's understanding of it. The framework comprises three interacting components: the designer, the user, and the system. Behind each of these are:

- The designer's model – the model the designer has of how the system should work.
- The system image – how the system actually works is portrayed to the user through the interface, manuals, help facilities, and so on.
- The user's model – how the user understands how the system works.

The framework makes explicit the relationship between how a system should function, how it is presented to users, and how it is understood by them. In an ideal world, users should be able to carry out activities in the way intended by the designer by interacting with the system image which makes it obvious what to do. If the system image does not make the designer's model clear to the users, it is likely that they will end up with an incorrect understanding of the system, which in turn will increase the chances of their using the system ineffectively and making errors. This has been found to happen often in the real world. By drawing attention to this potential discrepancy, designers can be made aware of the importance of trying to bridge the gap more effectively.

A more recent framework that has been influential in design practice is Garrett's (2010) user experience development process. Essentially, it is a software design life-cycle, depicted as a diagram (see Chapter 9 for more on these) that comprises five planes. These are the Surface Plane, the Skeleton Plane, the Structure Plane, the Scope Plane and the Strategy Plane. Figure 2.11 shows how they relate to one another, with the top plane (the surface) being the most concrete and the bottom plane (strategy) the most abstract. The framework also shows how each plane is dependent on the planes below it and that the concerns of each one become more concrete as it builds up. Such dependence reflects a ripple effect where decisions made early on affect those further up the planes. The framework has been used primarily to guide web development and understand the elements of the user experience. In contrast to Norman's framework of
design – that distinguishes between the designer’s and user’s understanding of the interface – it is intended to convey the bigger picture and, in doing so, help practitioners understand the context for the decisions they make.

In sum, paradigms, theories, models, and frameworks are not mutually exclusive but overlap in their way of conceptualizing the problem and design space, varying in their level of rigor, abstraction, and purpose. Paradigms are overarching approaches that comprise a set of accepted practices and framing of questions and phenomena to observe; theories tend to be comprehensive, explaining human–computer interactions; models tend to simplify some aspect of human–computer interaction, providing a basis for designing and evaluating systems; and frameworks provide a set of core concepts, questions, or principles to consider when designing for a user experience.

DILEMMA Who is in control?

A recurrent theme in interaction design is who should be in control at the interface. The different interaction types vary in terms of how much control a user has and how much the computer has. Whereas users are primarily in control for command-based and direct manipulation interfaces they are less so in context-aware environments and agent-based systems. User-controlled interaction is based on the premise that people enjoy mastery and being in control. It assumes people like to know what is going on, be involved in the action, and have a sense of power over the computer. For example, issuing commands based on a carefully designed set of syntax and semantics is often a very efficient and elegant way of performing many operations, such as spell-checking.

In contrast, context-aware control assumes that having the environment monitor, recognize, and detect deviations in a person’s behavior can enable timely, helpful, and even critical information to be provided when considered appropriate (Abowd and Mynatt, 2000). For example, elderly people’s movements can be detected in the home and emergency or care services alerted if something untoward happens to them that might otherwise go unnoticed: for instance, if they fell over and broke a leg and were unable to get to a telephone. But what happens if a person chooses to take a rest in an unexpected area (on the carpet), which the system detects as a fall? Will the emergency services be called out unnecessarily and cause carers undue worry? Will the person who triggered the alarm be mortified at triggering a false alarm? And how will it affect their sense of privacy knowing their every move is constantly being monitored?
Another concern is what happens when the locus of control switches between user and system. For example, consider who is in control when using a GPS for vehicle navigation. At the beginning the driver is very much in control, issuing instructions to the system as to where to go and what to include, e.g. highways, gas stations, traffic alerts. However, once on the road, the system takes over and is in control. People often find themselves slavishly following what the GPS tells them to do, even though common sense suggests otherwise.

To what extent do you need to be in control in your everyday and working life? Are you happy to let computing technology monitor and decide what you need or do you prefer to tell it what you want to do? How would you feel if your car told you to drive more slowly because it had started to rain?

Assignment

The aim of this assignment is for you to think about the appropriateness of different kinds of conceptual models that have been designed for similar physical and digital information artifacts;

Compare the following:
- a personal paper-based calendar versus a digital calendar (i.e. diary);
- a shared paper-based wall planner versus a shared web-based calendar.

What are the main concepts and metaphors that have been used for each (think about the way time is conceptualized for each of them)? How do they differ? What aspects of the paper-based artifact have informed the digital calendar? What is the new functionality? Are any aspects of the conceptual model confusing? Do you use a paper-based or web-based calendar? What are the pros and cons of the one you use?

Summary

This chapter has explained the importance of understanding and conceptualizing the problem and design space before trying to build anything. It has stressed throughout the need to be explicit about the claims and assumptions behind design decisions that are suggested. It described an approach to formulating a conceptual model and described the evolution of interface metaphors that have been designed as part of the conceptual model. Finally, it considered other ways of conceptualizing interaction, in terms of interaction types, paradigms, theories, models, and frameworks.

Key points:
- It is important to have a good understanding of the problem space, specifying what it is you are doing, why, and how it will support users in the way intended.
- A fundamental aspect of interaction design is to develop a conceptual model.
- A conceptual model is a high-level description of a product in terms of what users can do with it and the concepts they need to understand how to interact with it.
- Decisions about conceptual design should be made before commencing physical design (e.g. choosing menus, icons, dialog boxes).
- Interface metaphors are commonly used as part of a conceptual model.
- Interaction types (e.g. conversing, instructing) provide a way of thinking about how best to support the activities users will be doing when using a product or service.
- Paradigms, theories, models, and frameworks provide different ways of framing and informing design and research.
Further Reading

Dourish, P. (2001) *Where the Action Is*. MIT Press. This book presents a new approach for thinking about the design of user interfaces and user experiences based on the notion of embodied interaction. The idea of embodied interaction reflects a number of trends that have emerged in HCI, offering new sorts of metaphors.


Johnson, J. and Henderson, A. (2002) *Conceptual Models: Begin by designing what to design, interactions*, January/February 2002, ACM Press, pp. 25–32. This paper explains what a conceptual model is and shows how to construct one and why it is necessary to do so. It is very cogently argued and shows how and where this design activity can be integrated into interaction design.

Johnson, M. and Lakoff, G. (1980) *Metaphors We Live By*. The University of Chicago Press. Those wanting to find out more about how metaphors are used in everyday conversations should take a look at this text.

McCullough, M. (2004) *Digital Ground: Architecture, pervasive computing and environmental knowing*. MIT Press. This book presents many ideas, concepts, and frameworks for designing pervasive technologies. In particular, it discusses in depth the many new challenges confronting interaction designers and architects when working out how to embed information technology into the ambient social complexities of the physical world.

---

**INTERVIEW with Kees Dorst**

Kees Dorst is Professor of Design and Associate Dean Research at the Faculty of Design, Architecture and Building of the University of Technology, Sydney, and Senior Researcher in Design Studies at Eindhoven.
Please would you tell me something about your background and your current passion?

I was trained as an Industrial Designer at Delft University of Technology. I also studied some Philosophy before moving into design practice – when I realized that I kept thinking about design too much. I took up a small research position at TUDelft to investigate the way designers reach integration in their projects. That project later turned into a bigger PhD study comparing the two paradigms we use to describe and think about design: Rational Problem Solving (in which design is seen as a search process from problem to solution) and Reflective Practice (in which design is seen as a process of learning and reframing). In my thesis ‘Describing Design’ I argue that these two ways of looking at design are fundamentally incommensurable, as they are coming from very different philosophical roots. My design practice then moved into management and consultancy, as well as journalism. Currently I am working with a broad international network of researchers on the application of design thinking for organizational change.

Are there any particular findings or insights about the nature of design that stand out for you?

The work on design expertise has given me an idea of the impressive breadth of activities that we so conveniently label design: there are many different kinds and layers of design activities. I find it exciting that we are now at the point of understanding these much more deeply. That deeper understanding allows us to create a level of discussion that is much more precise, and also to transport elements of design to other fields. I am convinced that the introduction of elements of creative thought and action that have been professionalized within the design disciplines will revolutionize the way we create solutions to the problems we face in many different fields.

Can you give me an example of this?

We live in an increasingly complex and dynamic world, where traditional forms of problem solving are showing unforeseen limitations. Let me explain. Recent technological developments have landed humanity in a state of hyper-connectedness, where we find ourselves linked to innumerable other people. We are living in a brave new networked society, but the problems we face have become networked, too – to the point where the most important problems we face have become so complicated that they seem impervious to solution. Governments, institutions, and companies alike are struggling to come up with answers and are forced to reconsider their old problem-solving strategies that always worked okay in a well-ordered universe. They used to abstract from the details of the concrete problem situation, decompose and analyze it, and reach a conclusion in due course. But this strategy will not work at all for today’s problems: a tangle of relationships within complex and overlapping networks. Problems are intimately related to each other and are so dynamic that the world will have moved on by the time the formal analysis is completed. You can see this happen all the time: governments in particular are used to a hierarchical and purely analysis-based way of problem solving, and they seem powerless to deal with the complex issues we are facing today.

More and more, people are turning towards the field of design for help. Designers have been dealing with complex, networked problems that involve multiple stakeholders for many years. And they somehow have been able to come up with creative solutions that satisfy many of the relevant parties. Designers propose frames and ideas in a solution-focused manner, and test these proposals through experiments. This approach is called a solution-focused strategy, as opposed to the problem-focused approaches that are the basis for conventional problem solving.

Are there any tools or techniques for developing alternative or innovative designs that you’ve found to be particularly successful?

This is hard to say … What I have found in studying the way design expertise develops, is that experienced designers work very differently from novices. That has alerted me to the fundamental problem that severely limits the usefulness of many tools and techniques: while these tools and techniques are normally developed to support the professional designer, they tend to be rule-based – and experienced designers do not work in a rule-based manner. Thus professional designers tend to see the tools and techniques as alien and disturbing to their natural design process (cumbersome, wordy, bureaucratic). And they are absolutely right. Rule-based tools and techniques would be particularly useful in education and in the early stages of a design career, but not much beyond that. I think this is a real challenge for the academic community: we need to conceive of support for designers that is appropriate for their level of expertise and doesn't unnecessarily disturb the natural flow of their design activities. What would such a non-rule-based tool or technique look like? This requires tool builders to be clearer on what qualities their tools or techniques aim to achieve, what the scope of their applicability is, and demonstrate to the intended users that they are constructed with a close knowledge of the processes they are supposed to support.
What is the hardest part of designing?

For me, the hardest part of designing is dealing with its fundamentally dual nature: it is an open process of creation, that is also goal-directed … In practice this means that the designer, at any point in the project, has the choice of either a problem-solving approach or a solution-focused approach. Choosing a problem-solving approach might lead to unnecessarily limiting the scope of possible solutions; choosing a solution-focused approach might lead to a process that just spins out of control. The wisdom to choose well in a particular design situation comes with a lot of experience.

What does all this mean for interaction design?

Interaction designers can play a key role in the developments that are sketched above. Of all design disciplines, they may be the closest to having the skills and knowledge to deal with the dynamic and complex problems that we are confronted with. After all, interaction designers have always been used to dealing with dynamic relationships and complex scenarios – in contrast to, for instance, industrial designers, who have tended to focus more on the physical design outcome. This ability to describe, understand, explore, and create new frameworks and relationships is the key strength of design into the future.

The challenge for interaction designers will be to look beyond the current borders of their discipline, and re-contextualize their current abilities to meet these bigger challenges. In some of the leading companies and institutions (especially service providers, like banks and cultural institutions), we already see interaction designers moving into very strategic management roles where their core skills and knowledge are applied far beyond the reaches of the interaction design profession.