Subject name: Interactive Design

Subject code: SEG-14-13

Created By: Ali Akbary

Learning Outcome

Objectives of this chapter are: -

- Introduction understanding conceptualization
-

Chapter 2 Understanding Introduction

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 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?

It depends on what you are designing or building. Traditionally, interaction designers begin by doing user research and then sketching their ideas. In Agile UX, ideas are often expressed in code early in the design process. It is important to realize that having a clear understanding of why and how you are going to design something can save enormous amounts of time, effort, and money later on in the design process. 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. In this chapter we describe the steps involved. In particular, we focus on what it takes to understand and conceptualize interaction.

UNDERSTANDING THE CONCEPTUALIZING INTERACTION

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 (HUD), augmented reality, gesture-based, etc.

Figure 1 - GPS HUD

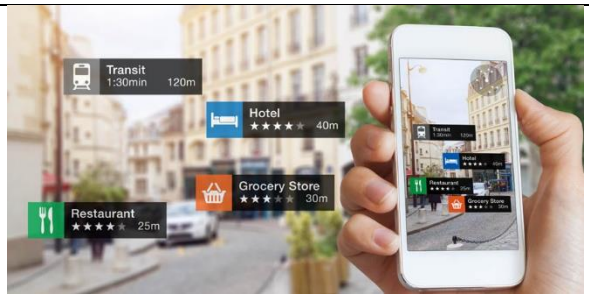

Figure 2 - Augmented Reality

The problem with starting here is that usability and user experience goals 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, 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. 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.

Figure 3 - Using GPS and Video on car dashboard

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,
- 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. Figure 4 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.

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 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.

Figure 4 - Scenario 1

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. Then do the same for curved TV screens. Are the assumptions similar?

Comment

3D TV went on sale in 2010 and curved TV in 2014. There was much hype and fanfare about the enhanced user experience they would offer, especially when watching movies, sports events, and dramas. An assumption for 3D TV was that people would not mind wearing the glasses that were 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. An assumption for curved TV was that it would provide more flexibility for viewers to optimize the viewing angle for someone's living room. But the unanswered question for both was: Could the enhanced cinema viewing experience that both claims become an actual desired living room experience? There is no existing problem to overcome – what is being proposed is a new way of experiencing TV. Were people prepared to pay more money for a new TV because of this enhancement? A number of people did. However, a fundamental usability problem

was overlooked: many people complained of motion sickness. The glasses were also easily lost, slipping down the back of the sofa. And wearing them made it difficult to do other things like flicking through multiple channels, texting, and tweeting (many

people simultaneously use second devices, such as smartphones and tablets, while watching TV). Most people who bought 3D TVs stopped watching them after a while because of the usability problems. While curved TV doesn't require viewers to wear

special glasses, it is not clear whether the claim about the enhanced viewing experience warrants the extra cost of buying a new TV.

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. The conceptual model is used by the design team as the basis from which to develop more detailed and concrete aspects of the design. In doing so, it can produce simpler designs that match with users' tasks, allow for faster development time, result in improved customer uptake, and need less training and customer support.

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 conceptual model 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 being 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 press the function Ctrl and D keys, 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.

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 are 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.

The four types of interaction are: -

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.

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.

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.

Activity

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.

Figure 5 - Two different types of vending machine

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.

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 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 to interact with a system in a way that is familiar to

them. For example, Apple's speech system, Siri, lets you talk to it as if it were another person. You can ask it to do tasks for you, such as make a phone call, schedule a meeting, or send a message. You can also ask it indirect questions that it knows how to answer, such as "Do I need an umbrella today?" It will look up the weather for where you are and then answer with something like, "I don't believe it is raining" while also providing a weather forecast.

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 . . .'

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 enable 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.

Many apps 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.

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. The basic 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) or play games (e.g. Minecraft). 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 2.8a); 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.

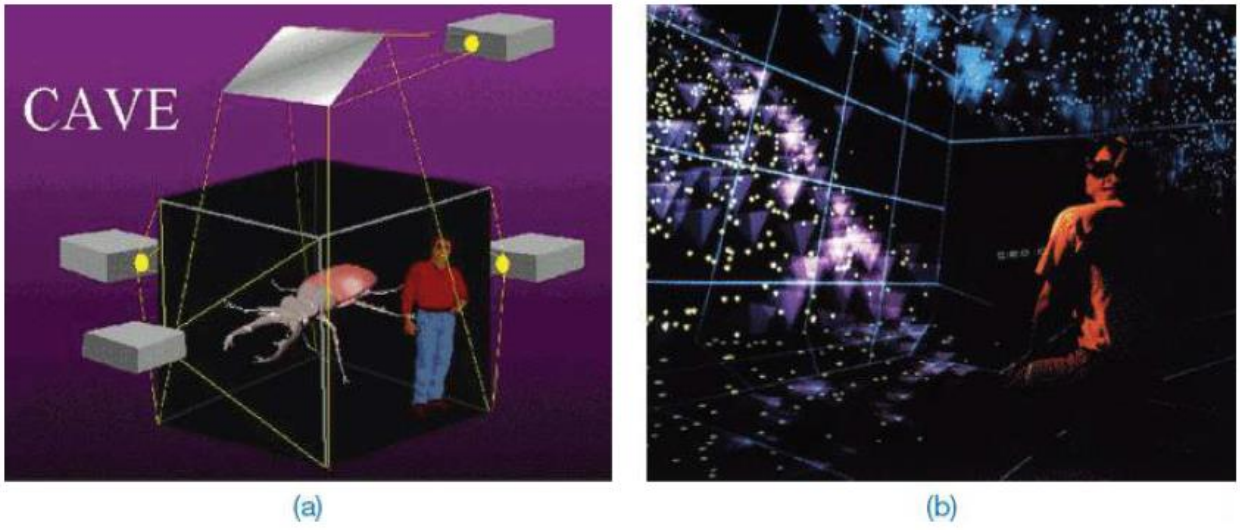

Figure 6 - (a) A CAVE that enables the user to stand near a huge insect, e.g. a beetle, be swallowed, and end up in its abdomen; and (b) NCSA's CAVE being used by a scientist to move through 3D visualizations of the datasets

A number of physical environments have been developed using embedded sensor technologies and other location-detection technologies. When the location and/or presence of people in the vicinity of a sensing device are detected, the environment decides which 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.

PARADIGMS, VISIONS, THEORIES, MODELS, AND FRAMEWORKS

Other sources of inspiration and knowledge that are used to inform design and guide research are paradigms, visions, theories, models, and frameworks (Carroll, 2003). These vary in terms of their scale and specificity to a particular problem space.

The five sources of inspiration and knowledge that used to inform design and guide research are: -

- **Paradigms** - 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.
- **Vision** - A vision is a future scenario that frames research and development in interaction design – often depicted in the form of a film or a narrative.
- **Theory** - 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.
- **Model** - 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.
- **Framework** - A framework is a set of interrelated concepts and/or a set of specific questions that are intended to inform a particular domain area (e.g., collaborative learning), online communities, or an analytic method (e.g., ethnographic studies).

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 were kept perpetually informed of what was happening around them, what was going to happen, and what had 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 has 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.

Visions

Visions of the future are another driving force that frame research and development in interaction design. A number of tech companies have produced videos about the future of technology and society, inviting audiences to imagine what life will be like in 10-, 15-, or 20-years' time. One of the most well-known is Apple's 1987 Knowledge Navigator, which presented a scenario of a professor using a touch-screen tablet with a speech-based intelligent assistant reminding him of what he needed to do that day while answering the phone and helping him prepare his lectures. It was 25 years ahead of its time - set in 2011 – the actual year that Apple launched its speech system, Siri. It was much viewed and talked about, arguably inspiring much research into and development of future interfaces.

A current vision that is driving much future technology development is the Internet of Things (IoT). By this is meant a scenario where people, objects, and animals are all connected through the Internet by having their own unique identifier. The assumed benefits of this kind of 'everything and everyone' connecting include improved services, up-to-date information and energy-saving utilities. An early example that has been much talked about is the smart home. Imagine your day starts with the heating/cooling silently turning on (and then off) to provide the perfect temperature for you while in the bathroom, followed by your alarm clock gently waking you up at the exact time you need to get up, while at the same time 'talking' to your coffee machine to start making the perfect cup of coffee at the time you want to drink it. Meanwhile, your fridge has sensed that you are running low on milk and fresh berries and has already sent an alert to your smartphone shopping app. As you walk into the bathroom, your smart mirror reveals how long each member of your family has cleaned their teeth for in the last week. You smile to see your son is cleaning his teeth regularly and for longer than all of you. Then you glance at the cat dashboard that shows a visualization of where your cat has been prowling the night before in the neighborhood. And so on.

These kinds of future visions provide concrete scenarios of how society can use the next generation of imagined technologies to make their lives more safe, comfortable, informative, and efficient. But they also, importantly, raise many questions concerning privacy, trust, and what we want as a society. They provide much food for thought for researchers, policy makers, and developers, challenging them to consider both positive and negative implications.

Many new challenges, themes, and questions have been articulated through these visions (e.g. Rogers, 2006; Harper et al, 2008), including: -

- 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.

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 (Rogers, 2012). 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.

2.6.4 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. 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 (2005) model of emotional design (Chapter 5).

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).

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 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 that 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.

In sum, paradigms, visions, 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; visions are scenarios of the future that set up challenges and questions for interaction design research and technology development; 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 sensor-based and context-aware environments, like the smart home. 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.

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 careers 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 will it feel to step into an autonomous car that drives for you? While it might be safer and more fuel-efficient, will it take the pleasure out of driving?