studio **7.5**

Designing for

Small Screens

Mobile phones | Smart phones

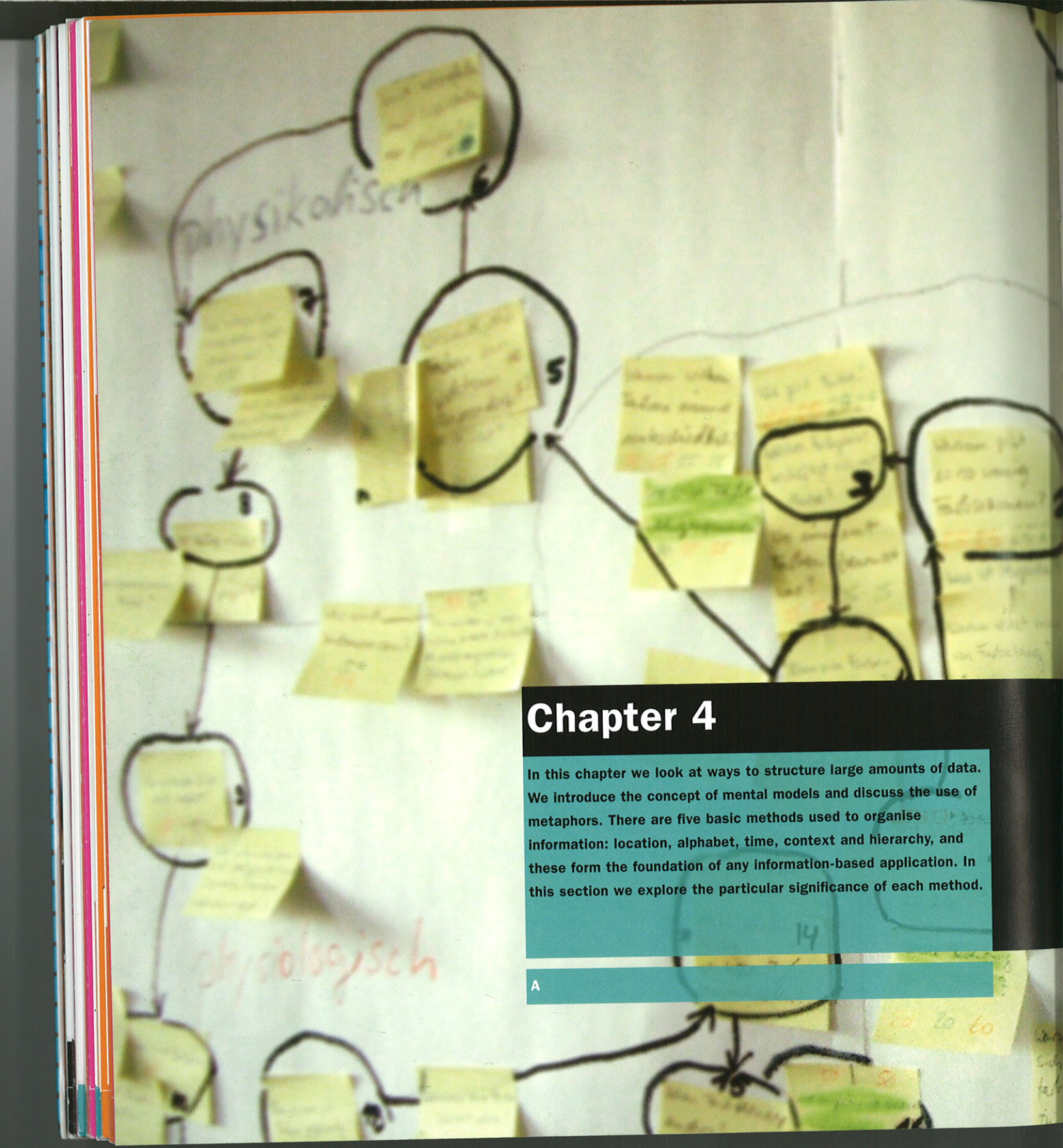
PDAs | Pocket PCs

Navigation systems

MP3 players

Game consoles

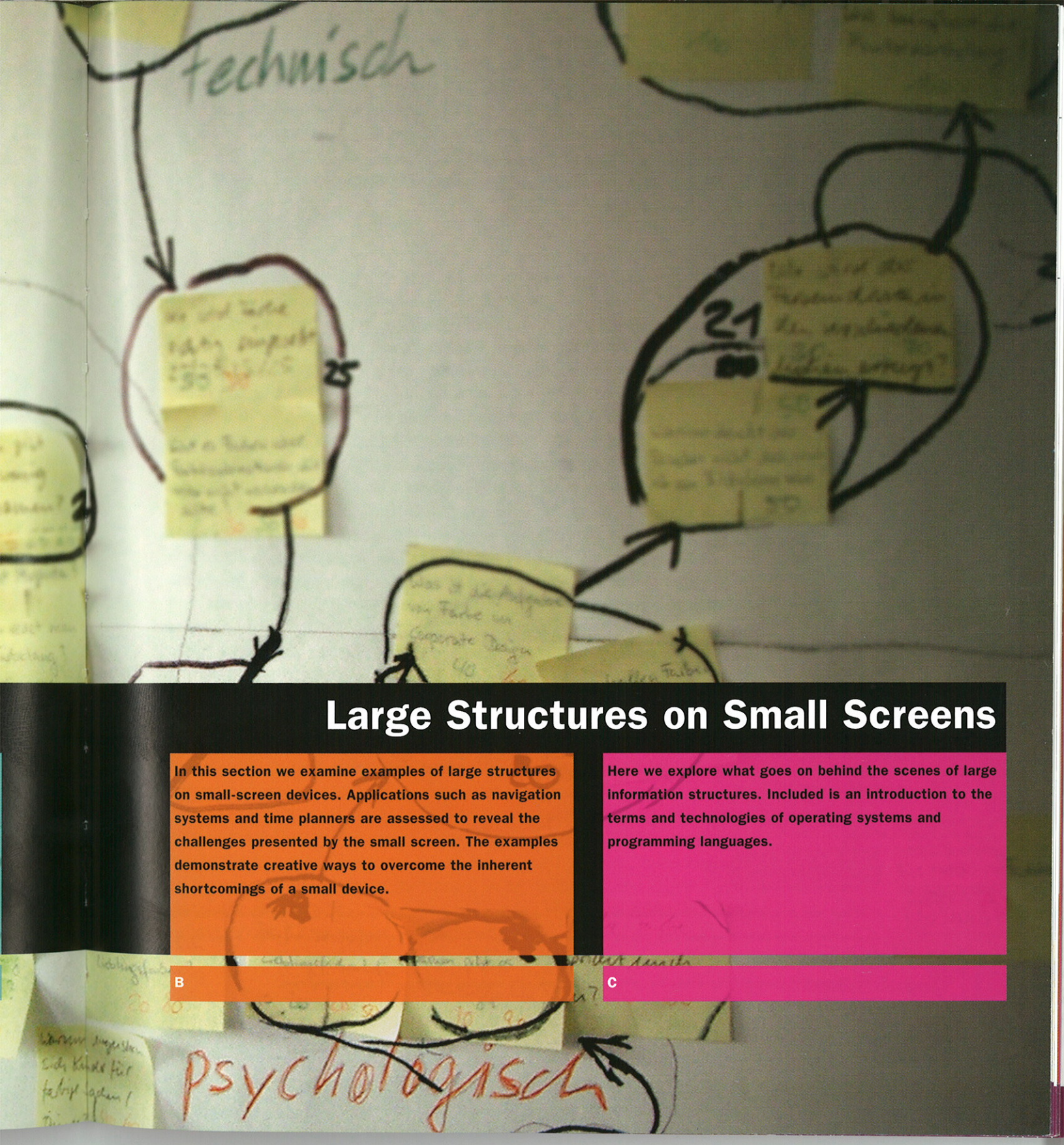
ava | Academia
the environment of learning



Chapter 4

In this chapter we look at ways to structure large amounts of data. We introduce the concept of mental models and discuss the use of metaphors. There are five basic methods used to organise information: location, alphabet, time, context and hierarchy, and these form the foundation of any information-based application. In this section we explore the particular significance of each method.

A



Large Structures on Small Screens

In this section we examine examples of large structures on small-screen devices. Applications such as navigation systems and time planners are assessed to reveal the challenges presented by the small screen. The examples demonstrate creative ways to overcome the inherent shortcomings of a small device.

B

Here we explore what goes on behind the scenes of large information structures. Included is an introduction to the terms and technologies of operating systems and programming languages.

C

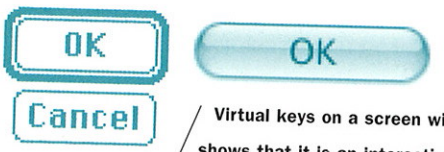
Large software structures can be very complex and extensive. The more generic the hardware is, the greater the number of applications that can be installed and used on it. However, this means that it is no longer possible to create an unambiguous link between the physical form, the interaction concept and the software application. Instead the interaction concepts must be made more general so that they can be used in a large number of different applications. The visual interface of the screen is all the more important because it must enable the user to understand the interaction concept for each program and provide the relevant operating instructions.

The mental model

It is important to convey a mental model that allows the user to visualise the 'invisible' space inside the device, this means that the user can gain an impression of the structure of the system. To create this sort of mental model the rules within the system must be plausible, easy to learn and the user must be able to rely on the rules once they have learned them.

Visualisations of the data in the form of maps and diagrams can serve as a basis for a mental model. This type of presentation enables meaningful links and logical interrelationships to be displayed – in a similar fashion to the results of a mind-mapping exercise. Later these models help with more detailed control because the user has already developed a basic concept of the overall system structure. For example, if a device needs operating instructions then some form of synchronous visualisation of these instructions should be employed, and ideally the main navigation level should offer the user any necessary explanations and instructions.

A mental model does not need to correspond with the actual structure of the software. Often, the user interface is a more simplified model of, or metaphor for, the processes that actually occur. Alan Kay, the inventor of the graphical user interface, calls this the 'user illusion'. The ability to form an appropriate mental model of the way a device works depends, to a large extent, on the user's experience in working with electronic devices.



Virtual keys on a screen will often create a tangible impression, thus reinforcing the link to the real world. This interface element shows that it is an interactive area that can be 'pressed' by the user. To simplify the interaction with the system, where there are several choices, it is sensible to predetermine the most plausible reaction for each situation, as this makes the confirmation process faster. This concept is important for small screens because it supports mobile, one-handed navigation.

See also Section A in Chapter 3: Physical Interaction.

< 050|051

The human ability to learn is characterised by the creation of a link with what is already known, and the transference of this knowledge to new situations. This principle is important as it means that the designer must know the intended target audience for a product when selecting its structure, in order to ascertain what likely prior knowledge and experience the potential user can call upon.

The **context-sensitive cursor** is a good way to give the user specific navigation hints or instructions. However, this feature is only possible if there is a pointer element for navigation.

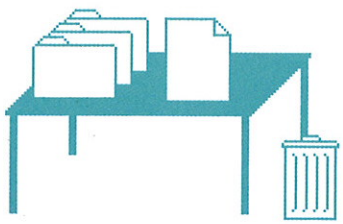
The **dustbin** is often used as a metaphor for the 'delete' function. This metaphor is so easy to understand that it has been in use for 30 years. It is sensible, especially in a drag-and-drop environment, because it embodies the instruction for interaction: if a file is dragged on to this symbol, it is deleted. It is also possible for the user to change his mind, because files can also be retrieved from the dustbin.

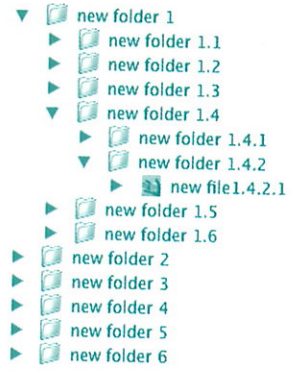
The metaphor is a practical form of the mental model. The metaphor, which is reflective of the human learning, starts with the familiar and transfers this to a new environment. The metaphor forms a narrative framework in which to place the possibilities within the system into a context that is logical for the user. Objects and actions are often borrowed from the real world to illustrate similar functions of the system.

The metaphor of the desktop, for example, enabled computers with this graphical user interface to be used for the first time by people with little or no knowledge of computers. The transfer of everyday objects and processes from the classical office environment and on to the interface allowed users to draw conclusions about the functions and the type of interaction that they offered. Equally, the interface of a pocket PC is closely based on structures that are recognisable from larger PCs, and as such it connects with the mental model that is already familiar to the user.

The concept of files that can be saved in folders, which can then be used to create new filing structures, has also become established in many systems. An **enclosing metaphor,** which defines the idea of the working environment in greater detail would actually be a hindrance, because it would transfer real space restrictions into virtual reality.

Metaphors can be a disadvantage. The metaphor as a closed narrative framework has not become established, and nor has a separate, all-embracing metaphor for small-screen devices. Instead, a number of single metaphorical interface elements have become established, such as the button, the folder, the magnifying glass or the dustbin. This is partly due to the fact that closed metaphors are difficult to scale, but a larger problem is the naturalism of metaphors: many functions of a digital system do not have any corresponding function in the real world, as such no metaphorical counterpart can be found.



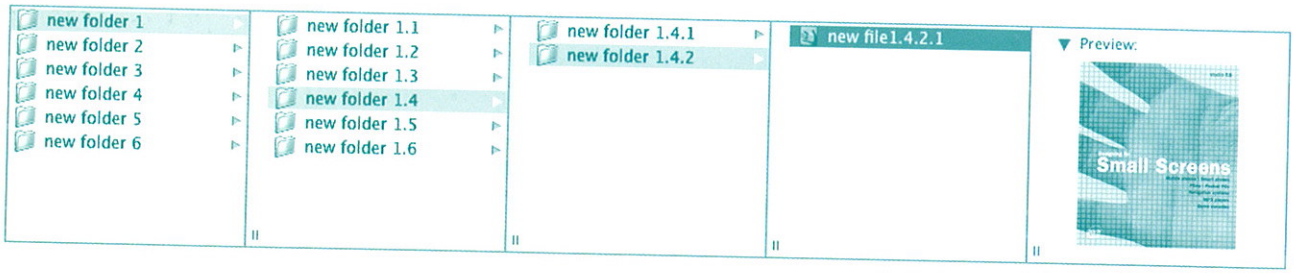


Nested lists work on small screens if the tabulator offset is kept to a minimum. This structure allows one-handed scrolling through all levels of navigation, without any need for a 'back button'.

Context

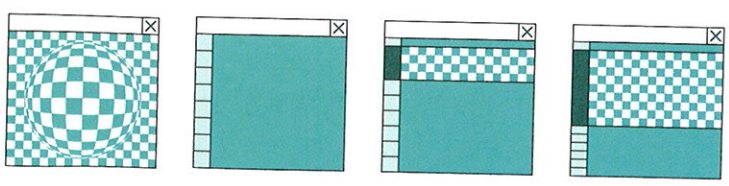
An important factor when considering orientation within large data structures is the preservation of context. The length of a book can be determined by the thickness of the its spine, however the size of a digital application is far more difficult to estimate. Equally if you look at any page in an open book, you can easily determine where you are within the progression of that book, but orientation within a digital application is far more difficult. The designer should therefore always try to give the user guidance and placement orientation. One way of doing this is to make several hierarchical levels visible at the same time. This gives the user a better idea of the structure and content as well as their position within the system.

Various display techniques have been established to show several hierarchical levels simultaneously. For example, a nested list is one way to show the tree structure of a hierarchy, or a preview of files in the form of thumbnails might correspond to a simultaneous display of the navigation and content levels. In the implementation of such concepts, small screens present a special challenge because of the limited available space.
A zoom function, especially a distorted zoom, can help to make economical use of the space.



Thumbnails enable files to be viewed without opening them. This is another concept that links navigation and content levels, and therefore makes interaction easier and more convenient.

On small-screen devices, techniques that **stretch or condense the content**, depending on the focus, can be used for long lists or large tables such as calendars.

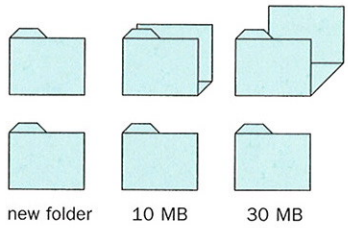


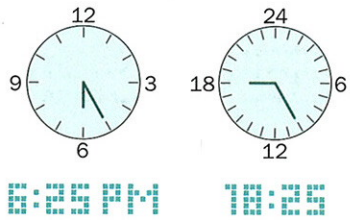
Analogue and digital

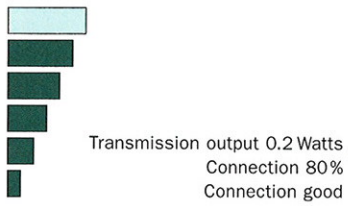
When numbers are presented on a small screen, an analogue display is preferable to a digital display. Diagrammatic visualisation can provide the contextual information that enables the displayed value to be interpreted. This applies not only to information that is familiar to the user, such as the time, but also to parameters with values and units which are less familiar or unknown. The numeric display of the current loading speed in megabytes per second is less helpful than a bar diagram that shows approximately how long the process will take and how far it has already progressed. The same applies to the signal strength, the charge status and the volume control. In each case, visualisation can show minimum and maximum values and thus display the current value in a meaningful context, which makes the interpretation easier and faster. Other applications that are suitable for analogue visualisation are file size, available memory space and call charges.

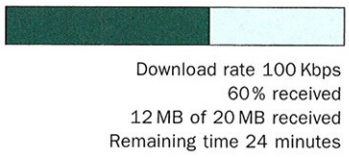
Only if display space is available should non-essential information, which allows a higher degree of precision, be added. For example, combining a visual display such as a loading bar and adding a time estimate can help the user to make a more qualified decision whether or not to download a file.

Visualising the abstract data, preferably within the range of values, creates easy-to-grasp information.

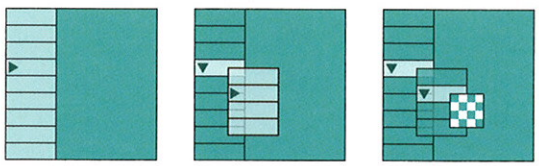








Preview options for the content can be displayed temporarily. This preserves the context and means that the decisions made on the next steps of navigation can be more informed. For other techniques to make maximum use of the screen space, see Section A in Chapter 2.



Organising information

According to the information architect Richard Saul Wurman, there are five different categories that can be used to organise information, these are: **location, alphabetical order, time** (sequence), **category** (context of meaning) and **hierarchy**.

In computer-aided systems, the designer does not need to decide which of these classifiers is the right one for the content that needs to be organised. The designer can instead use several organisational criteria, because the system is able to reorganise all data at any time and in accordance with any predetermined category. This means that the same information can be presented again and again, but in different contexts, and this functionality can help the user to find and retrieve specific data with ease.

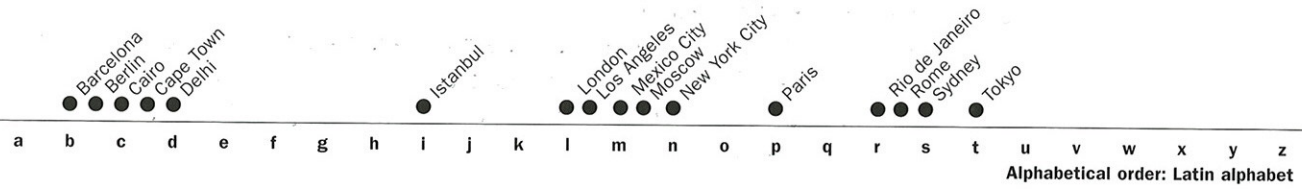
In the digital medium, redundancy is positive. The anticipated questions and expectations of the user should define the principles by which data is to be organised. When determining these, the designer can make use of the fact that the medium is able to respond to different user types and needs by supporting redundant navigation paths. In return, the designer must decide what meta-information should be included in a file.

These 16 cities can be organised by very **different principles**. Depending on the user's question, the data can be arranged in different ways. Sorting in alphabetical order is useful for large volumes of data, but it does not help to answer questions such as:
Where is Rome?
Which city is the oldest?
What language is spoken in this city?
Which city is the largest?
Which city has most inhabitants?

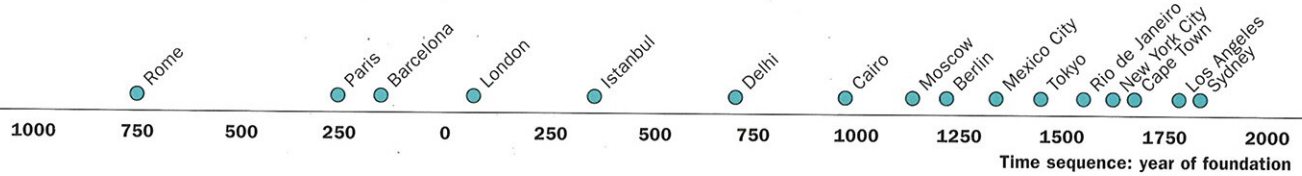
Location: geographical position

Basic design exercise

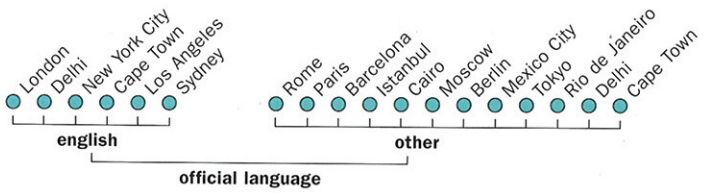
Develop an organisational structure for your music collection. Try to develop a system for each of the five organisational principles (LATCH), and then evaluate which of these concepts could be combined effectively for a more specific application.



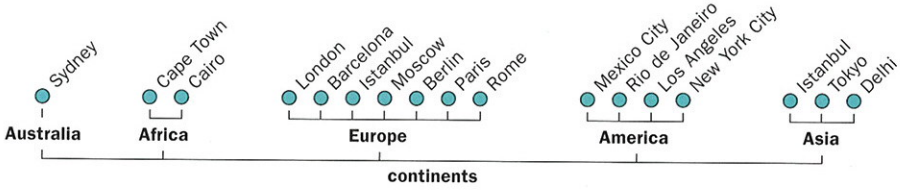
Alphabetical order: Latin alphabet



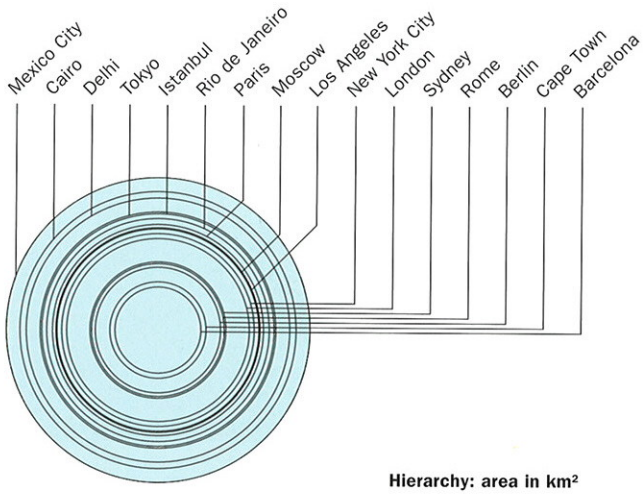
Time sequence: year of foundation



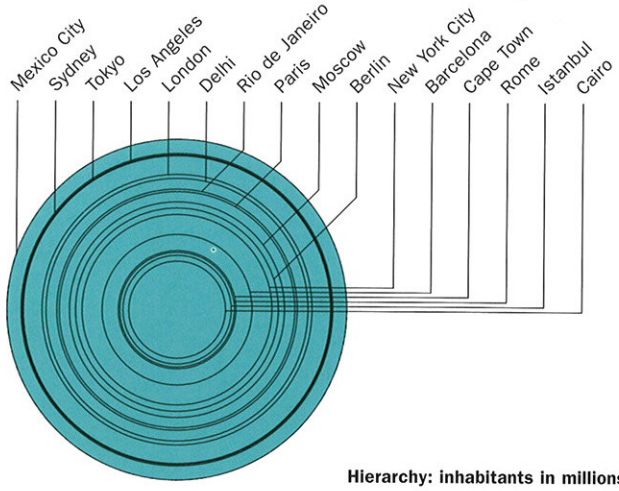
Category: official language



Category: continents



Hierarchy: area in km²



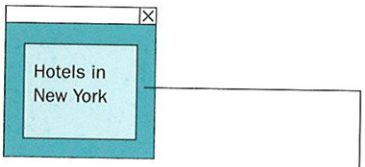
Hierarchy: inhabitants in millions

Location

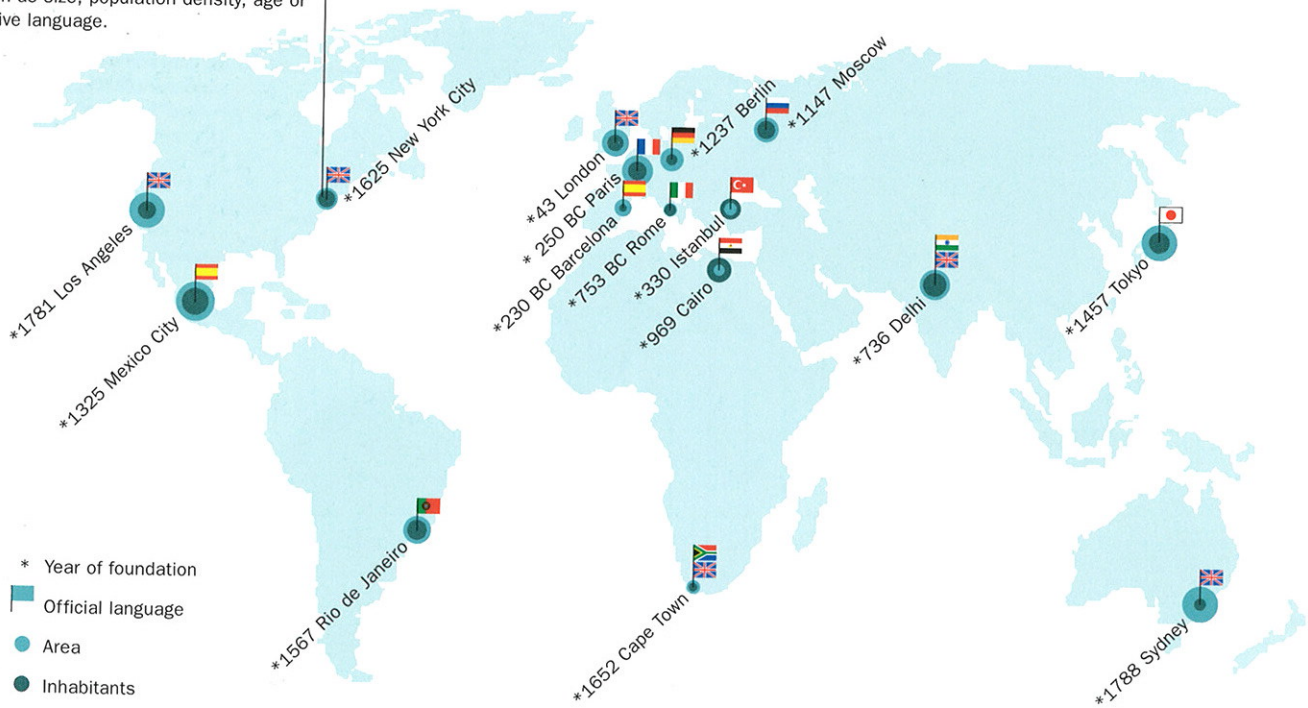
Sorting by **geographical** location is a powerful form of organisation, as maps offer a very efficient background on which other types of information can also be displayed.

The **real** location as an organisational criterion is increasingly important, especially for portable devices. The ability to find the location of the user and the device, for example, by a global positioning system, reverses the relationship between the hardware and the software – the user becomes a cursor in the real world. Location-based services use the information about the location to simplify the navigation within a software application. Hierarchical levels can be skipped because the geographical context information has already been evaluated. To ensure that this organisational criterion really makes things easier, the integration of the location must not be automatic, otherwise there is a danger that the choice of available options may be restricted.

Location-based services: navigation through the hierarchy is simplified by the fact that the device knows its own location, and includes this in the user guidance. For example, in New York, it will recommend products and services that are located in New York first.

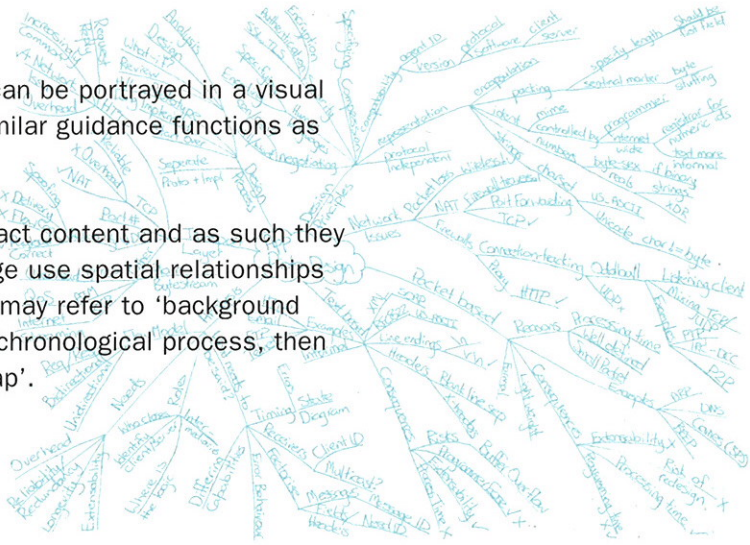


The geographical location can serve as a backdrop for all other information. It can form the mental model into which other information can be saved, such as size, population density, age or native language.



Complex data transactions and relationships can be portrayed in a visual form by **mapping**, and data maps can have similar guidance functions as those of a real geographical map.

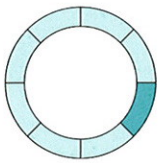
Data maps impose spatial situations on abstract content and as such they create meaning. Many concepts in our language use spatial relationships to explain abstract concepts, for example, we may refer to 'background information'. If the data mapping relates to a chronological process, then the mental model can even become a 'roadmap'.



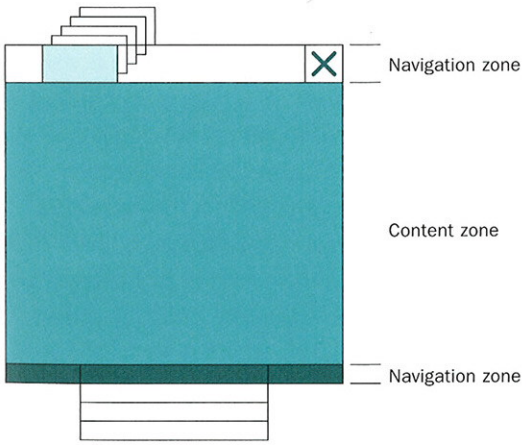
www.waitaki.otago.ac.nz

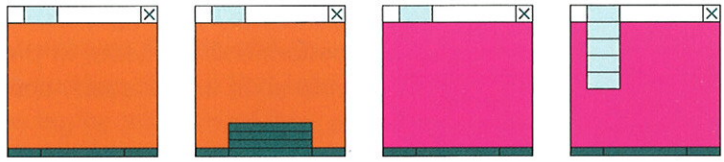
The **location on the screen** plays a decisive role in a very practical sense, as the display is a surface on which information is localised. The localisation of functions on the screen makes it easier to find them, and thus helps the user to develop a routine in their use of the system.

The designer can also make use of the user's spatial memory in the development of small-screen interface by subdividing the available space on the screen into specific functional zones and areas, and placing tools and navigation aids at the edges of the screen. Spatial memory can also be applied to unseen objects, such as pull-out menus or function bars. To fully make use of spatial memory, the basic functions must always be placed at, or pulled out from, the same point on the screen.

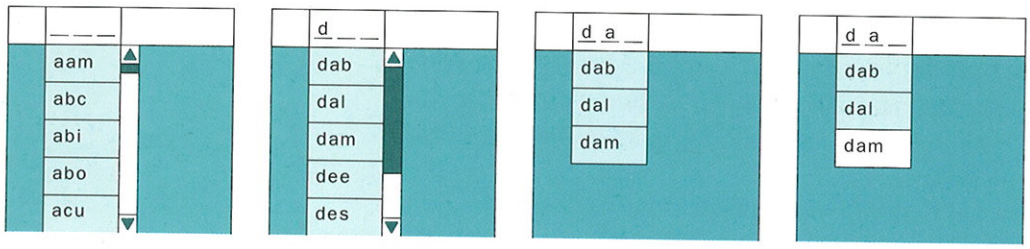


The circular arrangement of selection points is called a **pie menu**. It is much easier for the user to remember angles, rather than the position of an item on a list, as this takes full advantage of their spatial memory.





Human spatial memory should be exploited on small devices by assigning certain functions to a fixed place on the screen. The principle works even if these functions are temporarily pushed off the screen.

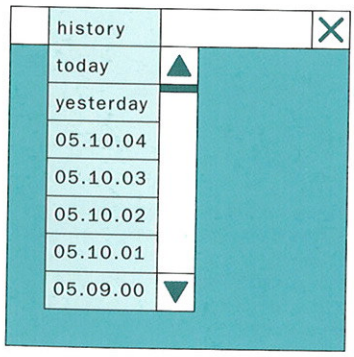


The principle of **reverse selection** offers the fastest navigation in alphabetical lists.

Alphabetical order

Sorting by alphabetical order is a robust form of organisation. It enables a large number of elements to be made available in a sorted list, and is used particularly for sorting names and lexicographical collections. It is especially useful on small-screen devices to structure long lists via the option to pre-select the first letter. To increase the ease of navigation, it should also be possible to scroll the alphabetical list continuously. If the form of interaction is limited, for example, in small mobile devices or navigation systems in cars, the input can be simplified by reversing the selection principle. For each letter entered, the scrollable list becomes shorter and shorter until the selection is so small that the desired term or name can be seen.

The **starting point** for all chronologically-organised data should be the present.



If the **time sequence** is arranged vertically, the latest entry should always be placed at the top. This ensures that any new entry is visible immediately and long scrolling is avoided.

Time sequence

Organisation by time sequence is an important navigation aid in computer-aided systems. It is essential for users to be able to trace their activities in virtual reality by reconstructing or retracing their actions on the basis of the data trail that they created.

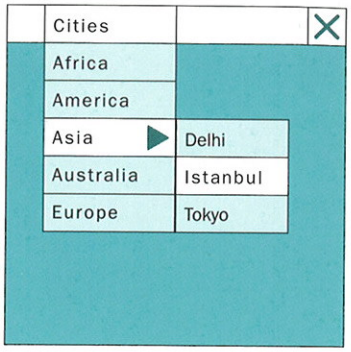
Time sequence can easily be displayed in the form of a list; time context requires a graphical presentation that indicates periods of time and the duration of events. When ordering events by time, the reading direction on the horizontal axis is from left to right: the past is found to the left, and the future to the right. On the vertical axis, the past is viewed at the bottom and the future at the top. In both cases, the present should be the zero point on the scale, and it should be visible when the application is opened.

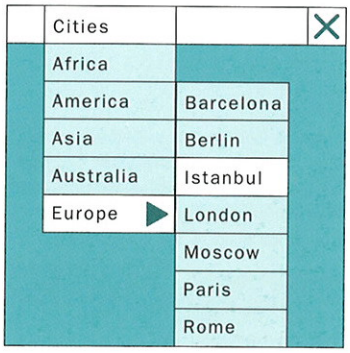
Category

Organising the content by meaning context is the most subjective of the organisational principles. Here you will find the greatest potential for discrepancy in the perception and evaluation of the content, and there is often a difference between the ideas of the developer and the ideas of the user.

Due to the restricted space on small screens, this discrepancy can force the user to spend a frustrating amount of time looking for their desired function because the structure is split into more hierarchical levels than on larger screens. Organisation by meaning context, which is often the main level of navigation for a software interface, should always be developed in conjunction with potential users or, at the very least, the concepts should be tested with the target users. In familiar applications, it will be important that the subdivision of the content matches the user's already established mental model. In unfamiliar applications, the basic organisational context must be communicated clearly so that the user can create an adequate mental model.

Organising the content by meaning context supports a user's spatial memory, as the concepts developed are often familar to those found in the real world, for example folders and filing systems.





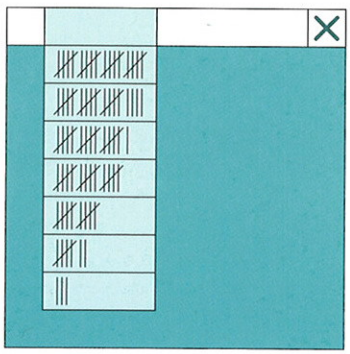
Redundancy is positive: in digital media, the data assignment may be simplified. If the cities are organised by continent, for example, a special case such as Istanbul can be listed under both categories without creating confusion for the user.

Hierarchy

The most common form of hierarchical order is by importance. This organisational principle is however, very subjective, and therefore it can cause the same problems as does arranging the content by meaning context. However, in the case of hierarchical organisation it is possible to derive the user's importance for each item via the frequency of use, and thus change the order dynamically. This principle is very suitable for small-screen devices because it helps to make effective use of the limited space and shortens the navigation paths.

If the relevance of a function is derived by other criteria than the frequency of use, the underlying algorithms can only ever be very incomplete approximations of the complex behaviour of the user. By making small assumptions, however, ease of navigation can be improved, and this is demonstrated by the results listings of websites such as Google and Amazon. Google sorts the search results by the number of links that refer to a site, and Amazon derives patterns from the buying activities of its customers by comparing users with similar interests.

The **hierarchical order** changes with use. The more often a function is called up, the more prominently it is displayed.



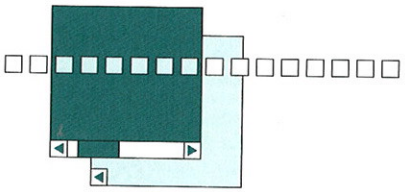
Shallow hierarchies

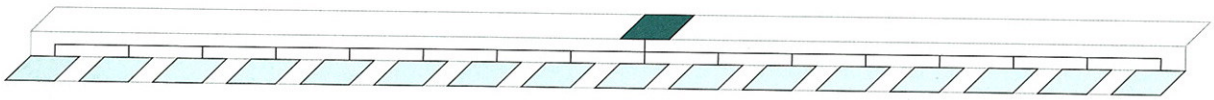
A decisive factor in determining the intuitive use of an application is whether it is possible for the user to gain a general overview of the system.

Shallow hierarchies are a good way to achieve an overview because they permit a synchronous display of all options available to the user. This is especially relevant for subjects in which the user may find it difficult to follow the logic of the organisational criteria. In such a case, the parallel display of all options can help the user to identify what they are looking for because there is no need to guess which category the term comes under.

However, this advantage of a shallow hierarchy can become a disadvantage if the number of options is so great that they cannot be displayed simultaneously or if the display becomes too confusing. The danger of overloading the interface is one of the main problems in adapting shallow hierarchies to small-screen devices.

In this example of a **shallow hierarchy**, the user moves directly from the navigation level to the content level.





Planning the hierarchy and deciding how steep or shallow the navigation structure should be is a task that the designer should tackle in conjunction with all other disciplines involved in the development process. The hierarchy determines the structure of the underlying data structures, such as databases and searches. It also has a great influence on the design options of the interface, for example, which navigation elements must be available at which point.
Also refer to the planning and visualisation techniques for complex structures in Chapter 9.

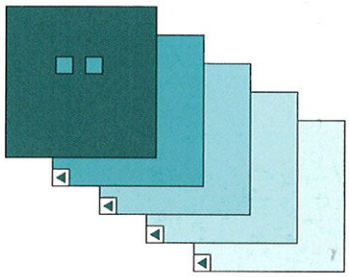
156|157 >

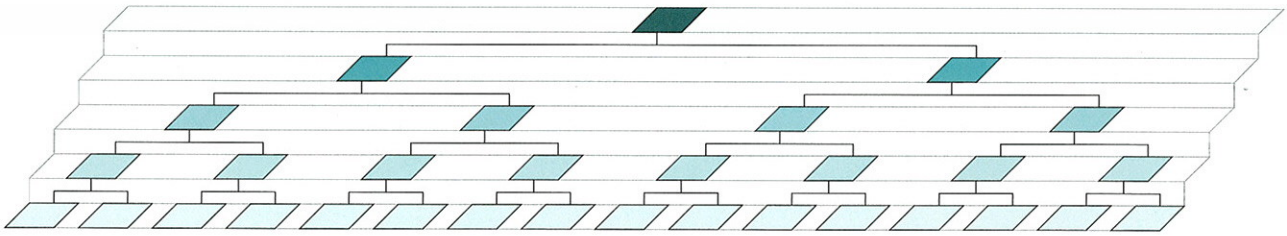
Steep hierarchies

Steep hierarchies break down a subject area into a branched structure that is organised by categories. An advantage of this model is that it offers a limited number of options on every level of the hierarchy. On each level, the user must decide which branch of the tree they wish to progress to. The selection is confirmed to move down through the hierarchy. To move back upwards through the structure, a 'back' command is needed.

A steep decision-making pyramid with a branch-like structure is difficult for the user to navigate in complex applications because it can be hard to understand the organisation of the content, and this may lead the user to guess whether the desired content will be found under a certain category. This necessitates a great deal of clicking in order to look inside the available categories, and a 'back' command is required after every wrong guess. This is partly because the hierarchy that a software development engineer considers appropriate may not be in accordance with the user's priorities and their subjective ideas of a hierarchy.

In this example of a **steep hierarchy**, the main navigation level leads to subcategories, and the content within these is then broken down into several further levels.





Advanced design exercise

Try to organise all the members of a particular species of mammal.

Is this family-tree hierarchy suitable for use as a structure for an interactive program?

Are there alternative criteria for sorting the animals, that could make a search for a species easier, such as habitat, size or colour?

Calendar Function on Mobile Phones: the presentation of calendars in small spaces is more or less standardised. The linear progress of time is broken down into separate views, such as months and days, but some of the contextual information is lost in the process. Periods that last longer than about two months are difficult to visualise, and so they are difficult to manage. Equally, showing appointments at the monthly 'zoom level' is a challenge for the designer.

Time-based arrangements are most easily displayed on small-screen devices in the form of tables. The annotations and tables for days, months or public holidays are familiar from printed diaries or calendars and therefore easy to use. The meaning context of years, months, days and hours is so straightforward that navigation between each of these 'zoom levels' works even without further explanation. However, this presentation breaks the time continuum down into discrete steps, which can make it more difficult to present periods and processes.

HP iPAQ RX3000

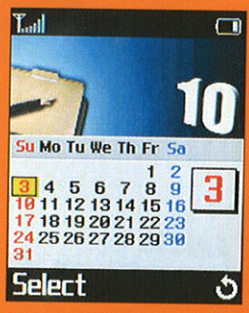
Panasonic P252IS

Samsung P730

Samsung P730

Palm Zire 21

Nokia 7270

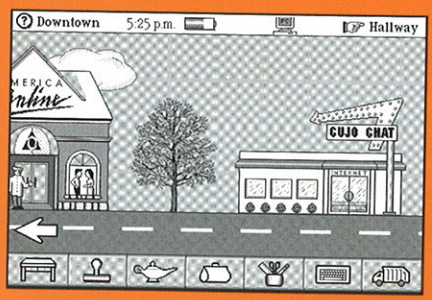


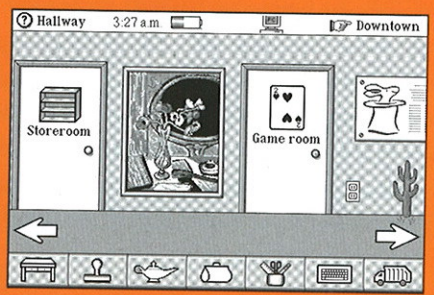
Samsung E610

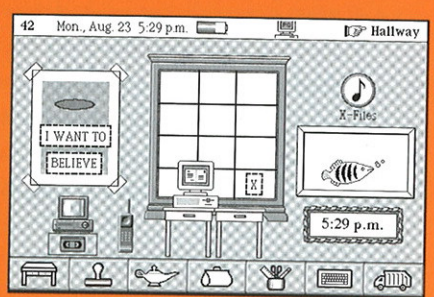
Samsung E800

Sony Ericsson K700

Philips 755





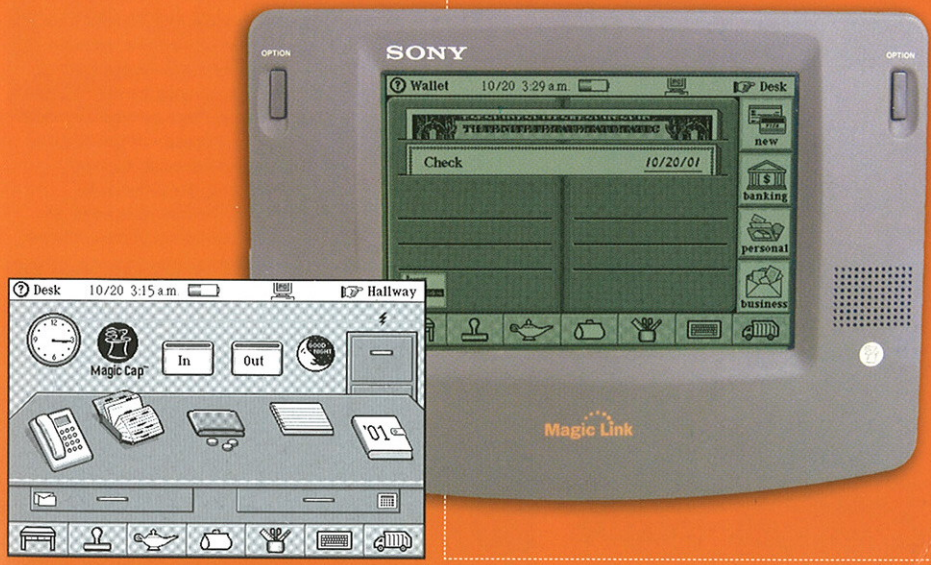


The ability to **link information about time and location** is a distinct advantage of the digital medium (compared with a conventional desk diary), but synchronous visualisation of this information on small-screen devices is a major challenge.

58 mm x 77 mm

Activis Scheduler: this concept for an organisational tool enables appointments to be planned and administered in their spatial context. All appointments that are entered in the vertical time bar are also shown in their spatial relationship to each other on the map. This means that the travel can be efficiently planned by ensuring that appointments that are spatially close together are also timed closely together.
(Hendrik Rieß, 2005)

Embedding complex structures into closed **metaphors** is a useful way to support intuitive use. However, this form of closed narrative framework is difficult to scale. This framework is limited when there is a large volume of data, and it is difficult to expand, so new data cannot usually be integrated in a logical way.



Magic Link PDA/Magic Cap Software (Sony, 1993): this device extends the classic desktop metaphor to include additional functions, such as the internet.

The metaphor is further extended by the addition of further locations, for example, the archive. There is also the possibility of going into town to shop. This metaphor is immediately understood by the user because it is based closely on reality.
(482 x 322 pixels)

58 mm x 77 mm

Applications are software programs that can be used on a computer, for example word processing.

Microsoft Office, Adobe Photoshop, Macromedia Flash, Apple Safari

The **operating system** administers all of the hardware components of a device, controls all processes and routines, and provides the user interface.

Linux, Mac OS, Windows NT, Windows Mobile, Symbian OS, Palm OS, Nokia IPSO

```
BASIC
PRINT "SmallScreen!"

C++
printf("SmallScreen\n");
```

A higher **programming language** works independently of the machine. The instructions can be formulated to address specific problems, and special programs translate the instructions into machine code. This translation can be done line-by-line by use of an **interpreter**, or before the program starts by a **compiler**.

Another distinguishing feature of programming languages is the underlying concept. The three classification concepts are **procedural**, **logical** and **object-oriented programming languages**.

Ada, ALGOL, BASIC, C, C++, COBOL, Delphi, Eiffel, Focus, Fortran, Lingo, Lisp, Modula, Objective-C, Pascal, Python, Visual Basic

Java, Smalltalk

Perl, PHP, SQL

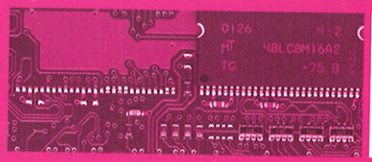
```
Basic's Assembler
ADR    R0, text
SWI    "OS_PrettyPrint"
SWI    "OS_NewLine"
MOV    PC, R14
.text
EQUS   "SmallScreen"
EQUB   0
ALIGN
```

Assembler is a **simple programming language** that is directly related to the physical hardware, such as the processor type on which the program will later be run.

0	0000 0000	32	0010 0000
1	0000 0001	64	0100 0000
2	0000 0010	127	0111 1111
4	0000 0100	128	1000 0000
8	0000 1000	129	1000 0001
16	0001 0000	254	1111 1110
17	0001 0001	255	1111 1111

Machine code is the binary code, or language, which is understood and implemented by the hardware.

8 bit = 1 Byte
1 000 Bytes = 1 KB
1 000 000 Bytes = 1 MB
1 000 000 000 Bytes = 1 GB
1 000 000 000 000 Bytes = 1 TB



Circuits form a physical part of the computer. Circuits translate the machine code into on or off signals: 1 = on and 0 = off

The distinction between operating systems and applications enables a range of functions to be extended flexibly; but this modular approach increases both the proportion of redundant processes and the risk of system errors and crashes.

As a result of the development of processor performance, an increasing number of systems are equipped with an operating system. Even small mobile devices are given operating systems that bring their range of functions close to those of a PC. This means that PC applications can now be adapted for small-screen devices.

Powerful programming languages, such as C, are used to write operating systems, applications or robot controls. As processors are becoming more powerful, these languages are increasingly used for microprocessors in embedded systems, although the code that they generate is more general and therefore less economical.

Some programming languages, such as Smalltalk and Java, use a 'virtual machine' for the execution of the program. This means that the applications are independent of the operating system and can be transported like normal files.

In addition to these powerful programming languages, there are also languages that were developed for special applications, such as SQL, which is used to program databases.

The direct adaptation of the hardware is especially economical, and it makes the processes fast. The disadvantage is the awkward programming, which can easily lead to errors. Today's processors largely compensate for the difference in speed between higher programming languages and assembler-generated code.

The hardware and software can interact in different ways in small-screen devices, and this affects the design possibilities. The development of portable devices follows the underlying concept of desktop computers, as such there is a trend towards modular concepts, which contain operating systems and applications.

PCs

Smartphones, Pocket PCs

PDAs, Organisers

Mobile phones

The further the interaction level moves away from the hardware, the slower and more error-prone the system becomes.

Game consoles

The more an application relies on the hardware, the slower and more inflexible the evolutionary development of the software becomes.

Traffic guidance systems, Logistics

Household appliances