Interfaces for ”dumb” users – can we hide too much?

Kai A. Olsen
University of Bergen and Molde College, Norway

Alessio Malizia
Universidad Carlos III de Madrid, Spain

Abstract

There is a tendency today, in the operating systems for PCs, tablets and cell phones, to hide the underlying file structure from users. The idea is to simplify by hiding “technical details”. In this paper we challenge this view. We show that there are situations where the user need to work on the file-level, and offer arguments that controlling the folder structure will simplify many operations.

Keywords: User Interfaces, Folder structure, Hiding.

1 Introduction

Modern computer-based devices, PCs, phones, tablets; etc. are designed for wide markets. Interfaces should therefore be intuitive, to be used by all. While most devices come with a user manual, this is often considered as the last resort. Users expect to push the on-button and start working right away. Reasonably enough, as these devices are acquired to make our lives simpler and perhaps more interesting. We do not expect to spend a lot of time to learn how to use them.

The techniques that are applied to attain intuitive user interfaces go in two opposite directions: hiding and visualizing. By hiding technical details we can reduce the number of objects and processes that an ordinary user needs to know. With visualization we can represent the more important objects in such a way that these can be handled by the user, for example through a natural interface using a touch display.

The same techniques are used for other types of technology. A car is a good example. Around year 1900 a driver would have to open the hood to pump gas, check the oil, perhaps clean the gas filter and carburetor, measure the amount of gas in the tank, and in many ways be his own mechanic. Today, with automation, drivers never need to see the engine. All the technical details are hidden under the hood. Few drivers check the oil, most rely on the lamps on the dashboard that give a warning if there is a problem. That is, the panel visualizes all that we need to know to drive the car. In addition to the warning lights we may see the current speed, outside temperature, tire pressure, and much more. There is no need to open the hood. In fact, there will perhaps be no advantage to have technical knowledge of how the car works. If there is a fault, which happens only rarely with a well maintained modern car, service personnel and specialized equipment are required.

Manufacturers of computer devices, operating systems and application software also have used the principles of hiding and visualizing for decades. The low level technical parts, the ASCII codes, disk track and sector numbers, program code, communication protocols, and system files are hidden from the user. At the same time care has been taken to visualize more high level objects such as programs, files and folders. Based on ideas from Xerox Parc in the late nineteen seventies the user is allowed to start a program by a click on an icon or “drag” a file from one folder to the other, i.e., operate directly on the visual representations of the

underlying objects. In this way, detailed control operations and complex syntaxes are avoided. This is also the underlying principle for the OAI (Object Action Interface) theory described by Shneiderman in 1997 [1].

With an explorer program, a file manager, the user is offered an overview of all data resources on the PC. All devices with storage capability that are connected to the computer are viewed as disks, with the folders and files most often presented as a tree-structure. While some folders, such as “My documents” and “My pictures” may be predefined, most users create and maintain their own folder structure. This will be role dependent. Some may be happy with storing pictures by date or by name of family member; a real estate agent will probably have a folder for each address while a tourist may use names of cities or countries for organizing photos.

However, on modern devices, from cell phones to tablets, we see a tendency to hide files and folders. The idea is to make things simpler by taking hiding a step further, using application dependent file locations. For example, when the user takes a picture the operating system of the cell phone will store this in a default location. Most users will not be aware of where images are stored. When the user want to view pictures this is done through a photo viewer that shows the contents of the hidden folders. A similar strategy is used for documents, spreadsheets, music and all other data objects. Some devices even come without an explorer program. Where folder and file names are created by the system, an explorer view will be of little or no use anyway.

The advantage of hiding files and folders under the “hood” is that the user can now think of photos, documents, spreadsheets, etc. instead of the more generic files. But there are some differences between an electronic device and a car. We will argue that the complexity, extendibility and flexibility of modern electronic devices make it impossible to hide the underlying data structure and that this approach will only confuse users in the long run.

2 Why hiding hurts

We offer six arguments against hiding files and folders:

1. There are situations where hiding files and folders are impossible. For example, to understand an important process such as backup, the user needs to have a concept both of devices and basic system processes such as copying.
2. Interconnectability also requires a basic understanding of objects such as devices, folders, and files, and processes such as moving and copying.
3. With more memory, broadband connection and improved functionality, specialized devices, such as cell phone, becomes full computer systems. The simple methods that were successful when organizing just a few objects becomes cumbersome when we get into large numbers of messages, photos or contacts.
4. Hiding assumes predictability. The system needs to know the aim of the user, what she wants to achieve. This is not always possible with a flexible and extendable machine such as a computer.
5. A generic view may in many cases be a simplification, for example to view images, documents, and audio as files.
6. Users and devices evolve together. Most users are going to use computer equipment for the rest of their lives and an investment in basic knowledge may be advantageous.

Each of these arguments is discussed in detail below.

2.1 When hiding is impossible

An application-oriented view of objects may work in many situations. For example, the photo viewer application can help the user to copy images from a camera, store these in default locations, and aid the user in finding, viewing and printing the images. However, this simplified view breaks down when one get device errors, e.g., an unreadable disk, a corrupted USB device or when we run out of disk space or memory.

Phone books on cell phones offer another example. These may be stored on the subscriber identity module (SIM card), in phone memory or on a separate memory card. Now, most phones will hide were the contact data is stored, for example, by using a default location. This works fine until the day the user buys a new phone. In order to transfer the contact information, the actual place were the data is stored suddenly become an important issue. In contrast to a car, we see that it is not easy to hide everything under the hood. But, of course, with a car there is no need to retain history and move data from an old to a new car (at least not yet).

Whenever a fault occurs in a car most users would have to call service personnel. However, in a computer setting even “dumb” users will often be able to fix many of the problems that occur themselves, perhaps with the aid of the operating system. An example is the “out of memory” situation, where the user can be asked to copy less or to delete uninteresting data before copying. Thus a basic understanding of the equipment is needed. As a minimum the user must know the difference between RAM memory and disk memory, what units that are connected to the system, how to copy, move and delete files and folders.

On most computer systems backup is left to the user. The consequences are seen in the news: lost master thesis due to disk error; laptop stolen with data that had taken months to collect; book manuscript deleted due to user error… Even if a system was set up with an all automatic backup, to mirrored disks and over a WLAN for external backup; the user would still have to know the basics. She would have to recognize the situation when one of the mirrored disks does not work and to understand that there would be no external backup when no network is accessible.

Apple Time Capsule may be used as an example. The idea is to automate backup and hide details from the user. The device takes a snapshot of the users system, using a wireless connection. Nevertheless, the device is subject to the problems described above. The user has to identify and handle situations where the wireless connection is down or when running out of disk space. That is, backup requires a basic understanding of how a computer system works, i.e., on concepts such as internal and external devices and folders. Devices such as the Time Capsule can simplify the task but cannot free the user of responsibility.

2.2 Copying data between devices

A user may have several devices, such as a phone, tablet, camera, USB key, etc. that can be connected to a PC. The traditional strategy has been to present all these devices as generic disks, with similar folder structures as on internal devices. The user can then view this structure through the explorer program, and may use standard desktop operations to move or copy files between the devices.

The alternative, that many modern operating systems use, is to let an application pertinent to the device type take care of the connection or to let the user choose among applications. For example, when a user connects a Nikon D90 camera to a PC running XP, the following alternatives are presented as default: view, edit and print, i.e., some of the basic operations that one may want to perform on photos. The problem here is that most users may want to

copy the pictures to a folder on the PC, but this operation is not among the options presented. The idea is probably to hide folders from the “dumb” user.

On the iPad, Apple use synchronization as the default mechanism. That is, when connecting an iPad to a PC the software will automatically synchronize data such as photos or documents between the two devices. The idea is to hide the copy operation from the user. But synchronization is a powerful mechanism that can have dramatic effects. The user may not want to transfer all the images from the PC to the tablet; there may not even be storage space on the tablet for everything. Synchronization may therefore easily result in an “out of memory” error, setting the “dumb” user in a situation that he cannot recover from. That is, without having an idea of files and folders. It is possible to limit synchronization to only one folder on the PC (e.g., the folder that contains the photos for the iPad) but this again requires that users have an idea of files and folders.

If the user has more than one PC, for example an office and a home computer, she may get into real trouble. At home all photos from the home computer will be synchronized with the iPad, for example by copying automatically all the new pictures from the PC to the tablet. But if she later on connects the iPad to her office computer, for example to synchronize documents, all pictures on the iPad that are not on the office PC will be deleted. The user can avoid this by turning off picture synchronization, but it would perhaps be smarter to let her perform the action that she needs, copying files to the iPad, than to ask her to remember to stop a set of automatic actions.

2.3 More resources

SMS (Short Message Service) have become popular, especially in Europe. The library sends a SMS when the book that you ordered has arrived; the airline tells you that your plane is on time; and we get a lot of messages from friends and relatives. Most cell phones store these sequentially in a default inbox-folder, similar to emails. However, while email system allows the user to create own folders, most cell phones have only this one folder. That works fine if there are few messages, but when these run into many hundreds it becomes difficult to navigate. A full explorer program with the possibility of organizing messages in folders would have been welcome. Instead we see that manufacturers move in the other directions, trying to hide folders.

The first cameras on cell phones had low resolution and offered a quality which was acceptable for viewing on the small phone displays only. In addition, available memory was very limited. Users were then happy to store a few pictures in the camera or send these to friends as a multimedia message (MMS). Today, with larger displays, high resolution, high quality camera units and an abundance of storage capacity, many use their cell phones as an ordinary camera and will also archive photos on the device. While sequential storing was sufficient when storage capacity restricted the number of pictures, we may now run into so large numbers of images on the phone that structured storage is required.

Files and folders would have been a natural solution in both of these cases, as on a PC. But since manufacturers want to hide the data structure from the user other methods are employed. Some systems use image annotation for organizing photos. This may be a replacement for a folder structure, but similar images may be annotated with different words at different points in time. Often it is therefore better, over time, to have a user-defined folder structure. We will, of course, have the same arguments for other types of data, such as documents, emails, spreadsheets, digital music and videos. The more capacity we have the more data will be stored and the more advanced data structure will be needed to manage all objects.

2.4 Predictability and flexibility

In order to simplify and hide details the user's aims must be predictable. For a car this will be the case, the uses of this device is clearly defined. But a computer system is much more flexible. Not even the designers will have a clear idea of what it will be used for. This is especially the case today where user may download new applications, modify software by the use of macros, add plug-ins, etc. To accommodate for this flexibility it is important to let the user control the data objects. Automation may simplify, but will restrict the way the device may be used.

For example, the owner of a small bakery that offers home delivery may get orders by SMS messages. Each message will list the number of products that the user wants, and the date of the delivery. The default chronological sorting of messages that his phone offers is not what he wants. To him it is important to store the messages based on delivery date. A more flexible explorer program could have offered this option. Alternatively, he could have developed an App, but this would also require access to the data objects.

Apple tries to hide the folder structure for MacBook users. A double click on a photo will start iPhoto. The image is then presented with an arrow button for seeing the next image. This works fine, as long as the next file in the folder is an image. If not, for example if the next file is a document, the right arrow will not work. Since the user navigates between pictures using iPhoto, a natural conclusion is that there are no more pictures in the file.

There are two solutions here: Either to increase the level of automation so that we can ensure that there is only one file type in each folder or to present the contents of the folder to the user. The first solution will impose rigid constraints on how the PC can be used. In fact, we may end up with a machine that can only be used as the designers intended. However, by presenting the underlying folder structure and letting the user navigate in this, the next button for the photo viewer can give the next image in the folder, skipping documents of other types. At the same time the user will be allowed to perform generic operations on the structure, for example to copy the folder to another device.

2.5 The generic view – files and folders

The advantage of letting a user work on low level objects such as files and folders is that on this level one may use a large set of "generic" functions, such as copying, moving, and deleting. Copying an image is of course similar to copying a document or a spreadsheet or any other data type. In some cases, for example when performing backup or deleting files to get more available storage space, objects are no longer images or documents but have to be viewed as just files, where the only interesting attributes are name and size.

As we have seen, by viewing external devices as generic disks instead of cameras, phones or music players we can perform operations on these, for example to move files, without learning new special operations. Any new type of device, a GPS navigator, an eBook reader, etc., can be handled in a similar way with the same operations.

2.6 The user evolution

Users evolve with the system. In fact we can talk about a co-evolution of users and systems. Nielsen [2] tells us that "Using the system changes the users, and as they change they will use the system in new ways". Designers present new devices and new applications, and users find new ways of using their devices which again force designers to make new versions and new interfaces [3, 4, 5].

This is especially the case for computer systems. We learnt this many years ago. In 1976 we made a simple data registration and statistical system that ran on one of the very first PCs. This was developed for doctors in primary health care that wanted to perform statistical research. It had functionality for constructing a registration form, for entering data and offered a set of simple statistical measures, sum, mean, variance, Chi-square, etc. After presenting the system at a medical conference we opened for questions and someone asked if this could be used for accounting. We patiently explained that this was a statistical system, that it could not be used for other purposes, especially not accounting. The answer seemed to satisfy the person asking the question, until another participant told the audience that he had used the system for accounting for at least a year and that it worked very well for this purpose. What he had found out, and what we had not seen as designers, was that it was simple to create a form for keeping track of bills and invoices, and using the statistical measures to get a sum for each account, albeit with values for mean, variance and the other statistical measures. Of course, in the next version we then offered better functionality for accounting. What is to be learnt from this story is that users may find new applications for a system that they master and, as important, where they have access to the basic operations and data objects.

The idea behind End User Development [6] is to present the user with a semi-structured system where she can contribute at different levels, from personalization to high-level programming. iGoogle (http://www.google.com/ig) can act as an example. Here the user is presented with many gadgets and is allowed set up her own, personalized interface. This is made possible by giving the user access to the data objects, in this case gadgets for news, weather, newspapers, date and time, etc. Developers can also make their own gadgets by using HTML and JavaScript libraries.

There is, of course, a cost benefit equation to user evolution. We cannot teach all users programming in order to let them be able to develop their own applications. But, perhaps we should let them be able to perform simple operations on the basic data objects?

3 Discussion

As we see hiding files and folders is in practice not possible. Sooner rather than later, users will get into situations where the need access to the underlying file structure or where such access will simplify operations. Still, many modern operating systems try to hinder direct access to files. In some devices users have not access to the underlying data objects at all, some do not offer an explorer program in the default configuration, and some will always let specialized applications handle data transfers, for example, by starting a photo viewer whenever a camera is connected.

We can ask why companies that have excelled in usability try to hide files even when the arguments to offer explorer functions are so compelling. The professional answer may be that they expect that large customer groups will only use the devices in the most simple ways (i.e., as planned, limiting the need for flexibility); that they have limited amounts of data (i.e., so that memory or disk overflow do not occur); that tablets and phones are connected to one PC only (i.e., to avoid synchronization problems); that backup routines can be performed by automatic systems and that users will contact service personnel whenever an error occur.

However, with access to the underlying data structure, looking at each device as a disk, we can exploit standard formats, for images, documents and for many other data types. It does not matter which camera were used to capture the images or on which computer system they are stored. We are free to move the data to other devices, for example, to the next PC that we want to buy or to the next cell phone. However, when files are hidden and specialized applications, such as photo viewers are used to access images these can now be stored in

proprietary formats. The customer may then be “locked in” to an operating system or to the devices manufactured by the one company. Perhaps this could be the rationale for hiding data objects?

Simplification is important, but should not be taken to the point where we make users “dumber”. Instead we have argued that users should evolve with the system. If this is to be possible some basic system understanding is necessary. This does not imply that we have to tell what is going on inside the processor, neither that we have to teach user the ASCII table. These things can be hidden without cost. But to explain that data is stored on different devices, that data is stored as files, to present the folder structure, let users name files and folders and offer a set of generic operations on these is necessary.

4 Conclusion

Today we see a tendency that makers of electronic equipment, from PCs to phones, try to hide the underlying file structure. We argue that this may have severe drawbacks, both with regard to how flexible the device will be considered to be, how efficient it can be used and in order to support the users in error situations. Most seriously is that it may restrict the free movement of data objects between devices made by different manufacturers, thus “locking in” the user to one company.

References

1. Shneiderman, B. (1997) Designing the User Interface: Strategies for Effective Human-Computer Interaction (3rd ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
2. Nielsen, J. (1993) Usability Engineering, Academic Press, San Diego, 1993.
3. Arondi, S., Baroni, P., Fogli, D., Mussio, P. (2002). Supporting co-evolution of users and systems by the recognition of Interaction Patterns. Proceedings of the International Conference on Advanced Visual Interfaces (AVI 2002), Trento (I), May 2002, 177-189.
4. Bourguin, G., Derycke, A., Tarby, J.C. (2001) Beyond the Interface: Co-evolution inside Interactive Systems - A Proposal Founded on Activity Theory, Proc. IHM-HCI 2001.
5. Carroll, J.M., Rosson, M.B. (1992). Deliberated Evolution: Stalking the View Matcher in design space. Human-Computer Interaction, 6 (3 and 4), 1992, 281-318.
6. Fischer, G. (2007). Meta-design: expanding boundaries and redistributing control in design. In Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction (INTERACT'07), Cecilia Baranauskas, Philippe Palanque, Julio Abascal, and Simone Diniz Junqueira Barbosa (Eds.). Springer-Verlag, Berlin, Heidelberg, 193-206.