Usability Principles and Usability Testing

1 User-centric design

1.1 Introduction

The design of a system must satisfy the needs of the users that are going to use it. Since the application is a tool that must serve the users, and that users must perceive as a good application, it is not enough for the application to make its work properly. It has to provide good feelings to the users.

Some of the important requirements the application has to fulfil are:

- You have to make sure that the user feels he or she is controlling the application.
- Learning curve should be as planar as possible: the user must require the minimum possible time to learn how the application works. This can be achieved by avoiding unfamiliar behaviours from (known) menus or buttons.

We can only achieve this if we have an idea on the user, his profile, and his behaviour. Therefore, user interface design is related with other disciplines such as psychology, ergonomics, and so on.

1.2 User centric design

Several studies have concluded that left-handers tend to die younger than right-handers. There are different opinions, and some say that right-handers live from 2 to 9 years more than left-handers. Some people believe that left-handers live in a world mainly **designed for right-handers**, and this may be one of the reasons that make the difference.

When designing user interfaces, we have to concentrate on the potential users of our system, without restrictions. Taking into account the needs of a small group may benefit (or at least not generate problems) the whole amount of users. For instance, selecting colours that avoid problems for people with colour blindness does not hurt the other users.

The **user model** is the set of details that a user may have, from the point of view of user interface design. We have to analyse different aspects of the users that may determine how we design the interface, such as:

- **User's knowledge on computers:** If the user does not know how to use a computer, it may become difficult to perform a task.
- **The mental model of the system:** Having a clear idea on the objectives of the users' work, and the application expected outcome is essential to avoid erratic behaviour.
- **Physical and sensory abilities:** Different users may have different physical abilities. We do not have to produce Power Point slides that can only be read by people with perfect vision, but the letters must be big enough to make them readable by most of the people.
- **Cognitive abilities:** Different cultural levels or degrees of experience doing a certain work may make some users more able to work with a computer. Our application should be useful for novice or inexperienced users as well as more experienced ones.
- **Capacity of identifying and performing tasks:** Sometimes the application or the user may be very specific. While for critical applications the user will have had a proper training, web-based applications may be tested by users without a clear

knowledge on the application workflow, or if the application is suitable for the problem in mind.
- **Learning capacity:** Interfaces must be more informative if the user has lower learning capacities. For classical applications, where the user has a clear idea on the expected outcome, little information may be required.
- **Personality differences:** Shy people may not be willing to explore the interface, and therefore features may not be discovered.
- **Cultural difference:** Some aspects may or not be acceptable for different cultures. An example is the use of images of flags to identify a language. It may be controversial and therefore should be avoided if our potential users may be offended.

Other factors may help to define in a more adequate way the interface:

- Expectance
- Motivation
- Preferences

By studying the user model, we may improve the interface in different ways: For instance, we may have different versions for different user profiles (novice to expert users), provide accessibility functions, such as different font sizes, and so on. However, it is not easy to obtain a user model, and when we have such model, it may be very difficult to act accordingly in order to improve the interface. Sometimes the information gathered might include much *noise* that does not help us. Moreover, some conditions may change along time, such as the cognitive capacities, and therefore our application may require adaptation to the new conditions.

Modelling all the possible users may become problematic, but if we are able to identify groups of users, this information may be important to improve our product.

User interfaces may be designed in order to be adapted to different users. It is important to distinguish two different concepts:

- **Adaptability:** The possibility of modification of the application interface by the user.
- **Adaptativity:** The ability of a system to automatically adapt itself to the user.

In order to illustrate the differences between those, we may take the example of the Gmail application. This application is adaptable because we may change the look-and-feel and its behaviour (we may for instance create filters for incoming mails). The application is also adaptive because some features such as the spam filters or the Priority Inbox react to the users behaviours.

1.3 Tasks

We have to see a computer as a tool. Particularly, it is a tool that allows us to perform the tasks we already knew in a more efficient way. Most of the tasks we carry out on a computer (write a text, store information on customers, and so on), are common tasks that users had already been doing for a long time in a manual fashion. Thus, the user has acquired a way to work in order to achieve an objective that he or she knows has a number of steps.

When the user is in front of a computer, his tasks must reassemble the tasks he already knows. Thus, their appearance must be similar; their representation should also be similar, and so on. So, if in the *real world* a certain task requires three steps, these steps should be somehow represented in the work to be performed on a computer. Otherwise, the user must learn again new processes or to get familiar with new tools and command names.

Other aspects, more related with the physical or social environment should also be taken into account. This includes ergonomics. Applications must be designed in order to be portable between devices (this includes screen size changes, for instance), to support data interchange with other users, data safety, etc.

1.4 UI life cycle

The construction of any interactive system is usually carried out by a cyclic process that involves design, evaluation, and development. Design evaluation is very important towards the definition and proper fine-tuning user interfaces. Final users must be comfortable with the UI, since they are the ones who will use the application. Once the system is working, we can proceed to refine the interface by evaluating parameters such as response times and the adequacy of the feedback.

The user should be taken into account through all the process. We may see in Figure 1 the life cycle of a user interface and the intervention of the user in all the stages.

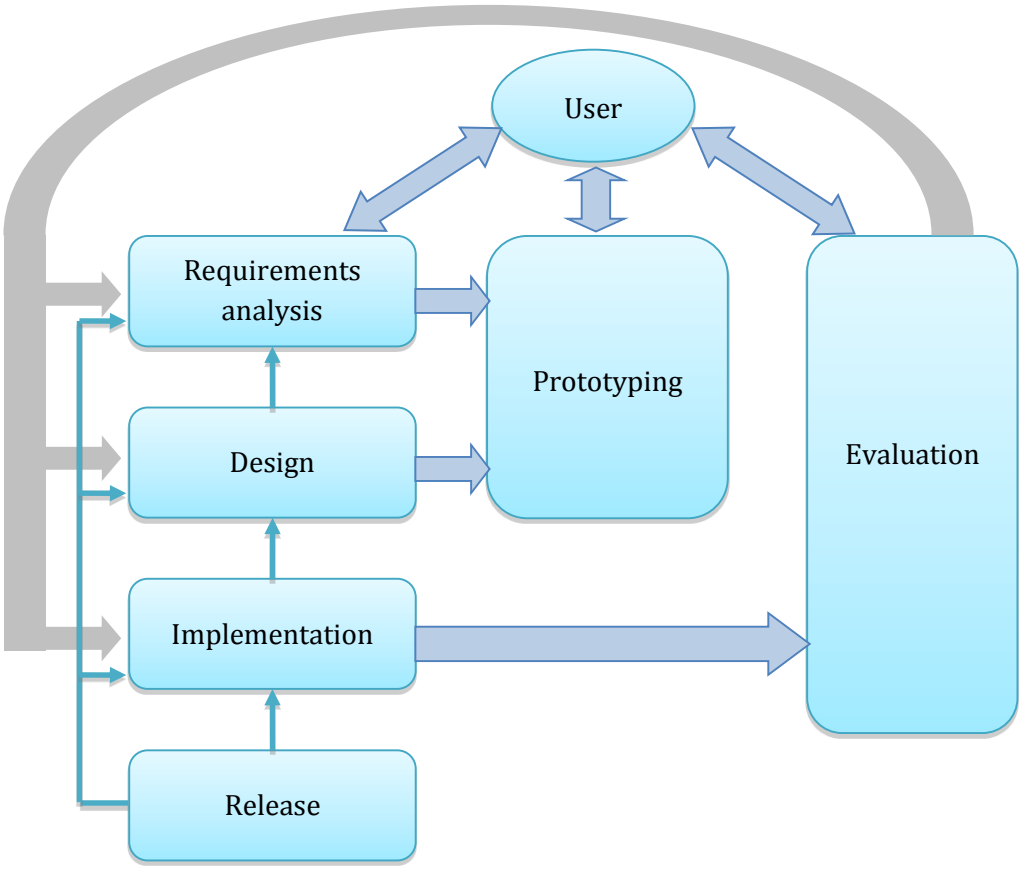


Figure 1: UI design. Participation of the user in the different stages.

1.5 UI design

Several years ago, the same person that analysed and developed the application carried out the design. Although this is less frequent currently, it still happens. But it has become more common to have professional designers for this task.

Designs must be focused to people with different abilities and knowledge of the application. Ideally, we must think both of novice users and power users.

There are three sources of information that designers may use in order to learn UI design: Principles, guides, and standards.

- **Principles:** They are general objectives that may be useful for the design. They give advice on how to proceed at a high level, but have to be adapted to the low-level problems we address in any concrete application.
- **Guidelines:** Design guidelines are a set of recommendations that should be applied when designing an interface. They are often measurable quantities (i. e. how many millimetres of spacing are enough for the proper reading of text, how many items are enough for a menu…). Guidelines must be general, and are often the result of extensive evaluations, together with the knowledge from ergonomics, cognitive theories, and so on.
- **Standards:** These are principles and guides that must be followed. Some standards are *de facto* (such as the ones imposed by important companies), and are used to protect the uniformity of the different products under the same OS, for instance. This has some benefits for the user, such as the familiarity with the icons and tasks. Other standards are called *de iure* and are the ones promoted by institutions like ISO (European), ANSI (EEUU), or AENOR (in Spain). Their objective is to establish a common set of rules in order to preserve uniformity (i. e. electric plugs…).

2. Principles and theories

Principles are fundamental statements that are widely applicable, although may require the adaptation to the concrete application. Compared to principles, theories go beyond that and try to explain user behaviour, user performance, perception, and many other problems. Their objective is to provide means to researchers to be able to understand relationships between concepts and generalizing results.

There are different levels of theories that analyse aspects such as the relationship between users and devices to theories that build a mental model of users of drawing programs. From the different theories that have analysed aspects of Human-Computer Interaction, one of them is highly relevant: GOMS. The GOMS (Goals, Operators, Methods, and Selection rules) model postulates that users address tasks using a four steps method:

1. First, the users define a goal or sub goal, such as inserting a word.
2. Second, the users start thinking in operators, such as the cursor keys necessary to perform the task.
3. Then, the users use the method, such as moving the cursor to the correct position.
4. Finally, the users use some selection method, such as the appropriate keys that insert a word, to finish the task.

Since its postulation, GOMS has been modified in various ways, but the details are beyond the scope of our course. Other theories model the behaviour of people, such as the OCEAN model, where OCEAN is again an acronym for openness, conscientiousness, extraversion, agreeableness, and neuroticism. These theories are used, not only to understand how the user works with a certain interface, but also to model the behaviour of agents in a life-critical situation such as emergency evacuations.

The principles commented here should be present on the designer's mind throughout the development of applications, especially when designing the UI.

2.1 Design for the people

We must keep in mind that the development of any application is focused to a certain group of people. Thus, the first we have to do before designing anything is getting to know the users and the tasks we want to provide.

We must know beforehand the group of users our application is addressed to.

In many cases, we will have three kinds of users: novice users, knowledgeable intermittent users, and power users.

2.1.1 Novice users

Novice users may vary from grandparents to children, and in between we may have first-time users that have experience with similar tools. Non-experienced users are assumed to have little or no knowledge on the task, while first-time users may be familiar with the processes and the interface concepts.

When designing for novice users we must try to overcome their inherent limitations providing, when necessary, instructions and online help. Vocabulary must be restricted to a small set of terms used consistently that are familiar to the users.

The number of initial actions should be also small, and simple. Once the novice users have successfully accomplished a certain number of small tasks, their anxiety is reduced; they build confidence and tend to have a positive attitude.

Errors must be informed properly when novice users make mistakes, and information on successful accomplishment of the task is useful and encouraging.

2.1.2 Knowledgeable intermittent users

Many users are expert in various domains or were experts but now use some systems only intermittently. An example of such users may be former power users that have been moved to other responsibilities in the company. They may have a clear idea on the task to solve but they have partially forgotten the steps required in the tool they are now using. Consistent menus and menu ordering together with guides to frequent patterns of usage may help them to rediscover how to perform the tasks. These users may benefit from context-dependent help to finish recalling the missing bits of information they require to complete the task.

2.2 Avoid limiting the user target set

As stated, when designing an application, we have to focus on our users. Therefore, if we are implementing an application for engineers, we would like that *all engineers* are able to use it, even those that may have some sort of problem or inability, or the ones that speak a different language. If we fulfil some requirements such as accessibility and internationalization, we will be able to provide a better service to a wider range of potential users.

2.2.1 Accessibility

Accessibility means to facilitate to people with certain inabilities the participation in usual activities. In our case, this means to allow the use of our program to persons that may have inabilities (motor or sensorial). Some examples are:

- Users that do not properly distinguish colours. That is, users with some sort of colour blindness. Depending on our colour selection, we may pose them some problems that may go from simple discomfort, to total inability to carry out their work. One common error is to put elements on our UI that can only be distinguished from each other from the colour, instead of using a second cue, such as shape, labels, icons…
- Users with hearing problems may have difficulties if certain messages are only transmitted through sounds. As in the previous case a complementary visual notification may be of great help, especially if the notification is critical.
- Users with limited movements may also be challenged if some task requires the use of predefined keys that are not placed close to each other, that is, which requires the use of two hands. It may also be a problem for elder users to expect a fast input, such as double clicking. These actions should be parameterizable.

These elements are only some examples that may facilitate the process of making software accessible. Other elements may be the use of voice for introducing commands or text.

2.2.2 Internationalization

Internationalization means to design a program in order to be used in different language environments. Localization is the process that consists of translating different textual elements such as dialogs, labels, etc. Applications can be easily translated if we design taking into account internationalization, for instance by having the textual messages grouped in separate files. However, it is not the only issue. We have to take into account other aspects such as the sensitivity with respect to a culture or politics. Therefore, there are some aspects that should be avoided:

- Drawings with flags or money.
- Maps with political frontiers or names of places that are controversial.
- Lists of countries that are not sorted alphabetically, unless this is a requirement of the application.
- Icons with animals.
- Icons only with hands or feet.

2.3 Application context

When designing an application we have to take into account its context. This implies changing the vocabulary to make it consistent with the vocabulary familiar to the users. For instance, hospitals will usually deal with *patients* instead of *customers*.

In some cases, we may take advantage of the users' knowledge to create metaphors that represent elements of the application. This has been repeatedly used in GUIs. For example, a folder icon usually represents a document container. Of course, the use of metaphors from real world should not be too *realistic*: In real world we do not store a high number of documents (or other folders) in a single folder, but limiting the virtual folder capacity would be useless and unreasonable. Mixing real elements with different tasks may also be confusing: the use of the dustbin as an icon with the ability of ejecting a CD may be confusing (note that nowadays, Mac OS X, although placed at the same position, dynamically substitutes the dustbin by a *eject* icon).

2.4 Strive for consistency

Applications must be consistent. Although this might seem obvious, this is one of the principles often violated in applications. Consistency means to keep the same names, commands, icons, and so on across the application. But it also consists in maintaining the familiar names and icons throughout all the system. If all the applications use the same icon for the *folder* metaphor, changing this icon in our application generates confusion and, at the end, requires an extra mental effort from the user.

Consistency with other applications allows the user to use his or her previous knowledge from other applications for the fast comprehension of the new application. Note that, in most cases, the user will spend more time in *the other applications*, and therefore will have a higher amount of exposure to *the other look-and-feel*. Apart from better and quicker comprehension, consistency gives the user self-confidence and a familiar environment. Consistency is also required in the implementation of commands: Commands with the same name should react the same way than in other applications.

2.5 Keep the user informed

The user must know in every moment what is happening to the application. This is achieved by providing enough information at the right moment. Users must never *guess* the state of the system or application or they will probably guess wrong. When an action is started, we have to provide with some feedback that indicates that the command has been received and that the application is working on it. This information can be visual, audible, or both. If a certain task requires a high amount of time, we have to inform the users as carefully as possible (through the use of cursor, progress bars, progress dialogs…). Error messages must be clear and use a plain language: They must provide the reason of the mistake and help the user to get out the situation if this is possible.

Accuracy is important in feedback. Progress indicators must be as much accurate as possible. If the time required to perform a task cannot be computed accurately, we have to indicate so. Generic messages make the system difficult to use.

2.6 Strive for simplicity

Large amounts of contents may distract or even disturb the user from doing his or her task. The application should allow the users to concentrate in their work. Therefore, only relevant information must be presented, together with the elements necessary for the interaction. Every extra element reduces the visibility of the other, really important ones. They also distract the users from the relevant tasks. Overloading the UI with unnecessary buttons, menus, or irrelevant information makes the system less usable. Advanced options must appear progressively.

Nonetheless, the way of presenting the information must be comfortable. Widgets must be properly aligned, following the usual direction of reading, and in a logical order. A short and harmonic colour set must be used to avoid visual discomfort and fatigue.

2.7 System under control

Computers are built to serve the humans, and not the other way around. A user must feel he or she is controlling the system. This implies that the system must respond adequately to user input: Tasks must be accomplished in a reduced time, and the user should not be blocked, because this makes the user to reduce attention. Modal windows are also a threat to this principle, so they must be avoided as much as possible.

If it is possible, the application must provide the tools that allow the user to configure the environment in order to make it more comfortable. However, too many configuration parameters, or making the configuration a requirement may prevent the user to work naturally. Obviously, parameter changes must be properly explained with simple and plain language. When possible, default values must be provided.

2.8 Pardon the user

We know everybody makes mistakes. It does not matter if you are an expert user or a novice. Our system must have a philosophy of service to the user. Therefore, we must avoid that an error destroys all the work a user has performed. One fundamental issue for achieving this objective is to allow undoing the work.

Moreover, we have also to warn properly the users if their actions are undoable, especially when those are dangerous. In these cases, we have to inform and to ask a confirmation from the user. It is important that the number of times the user is warned does not grow too much. In these cases, the user starts ignoring messages and confirming everything. This may have very bad consequences.

It is also important to handle with care the user's work. It is necessary to avoid situations in which large amounts of data or work can be destroyed by mistake, without an explicit action from the user. This can be addressed in many different ways: by saving the information automatically, by implementing undoable actions and from very different types.

2.9 Provide direct manipulation

Users tend to feel more comfortable when they are given the freedom to work as they wish. Therefore, it is usually a positive point to provide direct manipulation instead of a dialog

box. For instance, if an object must be dragged to a different position, it is good to let the user do it with the mouse instead of configuring the position using parameters.

2.10 Anticipation

Applications must anticipate user actions as often as possible. It is not a good strategy to let the users search for information if we can provide him or her in immediately. For example, a word processor can imagine that the following action of a selection will probably be a copy or cut of the text. Thus, a contextual menu providing most common actions usually saves time.

However, user guiding without his or her permission may become uncomfortable. A very annoying example is the option to automatically hid menu items without the user to know. In these cases, the user may unsuccessfully explore all the menus without finding the appropriate action. Such decisions must be coherent and predictable by the user.

2.11 Default options

Default options must be easy to change. For instance, if a dialog has default options, when the user puts the focus onto the option, the text must be automatically selected so that he or she can change the contents easily.

Default options should be *intelligent* and, like in the previous case, be predictable by the users.

2.12 Strive for user efficiency

The system must optimize user's work, not system's work. Usually, people's work is more expensive than machines' work. As a consequence, the system must avoid the user to be inactive.

When designing an application, the features that generate efficient workflows are typically architectural designs, not user interface designs. When designing the interface, we must care on the users.

For costly tasks, it may be useful to start them in background and let the user work in other things. If the user needs the action to finish, sometimes a partial result may be enough for the user to continue working, such as previewing images of lower resolution…

Menus can also be defined for efficiency: frequent actions must be more accessible. Large buttons and elements close to the borders and corners are easier to access, since the mouse movement in a certain dimension can be infinite. If we add a border around buttons placed near the limits of the screen, and this border is not clickable, buttons are more difficult to reach.

2.13 Plasticity

Plasticity means the ability of an application to adapt to several form factors (such as mobile devices and desktop screens) and screen sizes. With nowadays access to Internet and applications from a wide range of differently sized devices, our software should be capable to adapt to these different form factors.

This not only implies the ability of resizing the application window to the desired size of the screen's device. For small devices such as smartphones, the simple resizing will turn buttons and other input elements too small to be touched. Moreover, output will simply be unreadable due to the small size. Small devices should present an adapted version of the UI with larger buttons and smaller amount of information rendered using larger fonts.

The content must be also adapted, for instance avoiding costly flash-based animations (even if our device supports them) because they may drain the battery faster than expected.

2.14 Corollary: Don't make me think!

Like the title of the famous book by Kruger ([Kru06]), one general principle in design is to reduce the users mental work: Don't make me think. This improves usability and can come in different flavours: Use icons that are familiar to the users, be consistent throughout the application, and so on. In some cases this can be confounded as copying other interfaces' designs. This is not the case. For many years, designers have been working in the definition of icons that are related to different tasks of document management. As a consequence users are familiar with those icons, and changing them will make them think if the action under the new icons does exactly the same than with the well-known icon. This makes the user to waste time and feel unsecure.

Interfaces must be obvious, evident, clear and easy to understand.

3 Golden Rules

You may find different versions and an extended number of principles in different sources. Several authors have tried to reduce the amount of principles in what has been called a small set of *golden rules*. Some of them have already been mentioned explicitly, some of them group other principles. They can be used as a guide for designers, though they require, like principles, to adapt to the proper problem. We enumerate here with only little details:

1. Strive for consistency: Although it is difficult to follow, and often one of the most violated rules, it is very important.
2. Cater to universal usability: Recognizing the needs of different users is crucial to make our application accessible to everybody.
3. Offer informative feedback: System feedback must be provided for every action.
4. Design dialogs to yield closure: Users should not need to get out of the dialogs in order to complete them. All the necessary information should be provided.
5. Prevent errors: Input filtering can greatly reduce the number of possible errors in forms. Also, discouraged actions should be marked appropriately.
6. Permit easy reversal of actions: If actions are reversible, the user will reduce anxiety and work more comfortably.
7. Support internal locus of control: Users must be action initiators, not responders to actions of the application. Therefore, interfaces must properly respond to users' actions.
8. Reduce short-term memory load: The rule of the thumb is that humans can remember 7 +/- 2 chunks. Thus, UIs must be simple and reduce user's need to remember things.

4 Guidelines

Design guidelines are more concrete than principles, but are not as specific as design rules. They have been developed by several companies such as Microsoft and Apple. They provide a common language and promote the consistency among different interfaces by different applications designers.

They have been developed and improved with the results of various empirical studies and practical experience.

Guidelines are often stated with a series of do or do-not rules, that can be argued, and that are sometimes too specific. However, their use often improves the quality of the designs. We provide here some examples of guidelines that address some common tasks such as navigation, display organization or user attention.

4.1 Navigation guidelines

Navigation is a task that can be difficult for many users, and it can be a nightmare if the application or web we try to use has a huge amount of information and pages. Some rules from the National Cancer Institute, backed by research are:

- Standardize task sequences: Similar tasks should be performed by following similar steps.
- Ensure that embedded links are descriptive: When using embedded links, the description should clearly state the place we are going to bring the user.
- Use unique and descriptive headings: Headings must be consistent and clear across the application or web page.
- Use check boxes for binary choices.
- For web pages, develop them so that they print properly.
- Use thumbnail images for previewing large images: If full-size image viewing is not critical, this may save a lot of bandwidth.

When designing applications and webpages, we also must take care of different accessibility issues such as textual descriptions for elements in our UI that are not text, writing transcription for talks in videos, colour-blind safe designs, and so on.

4.2 Display organization

Display design is a main problem and has many special cases. Some guidelines that must be followed are:

- Design data consistently, including formats, colors, capitalization, terminology, and so on…
- Provide a format that allows for an efficient assimilation of the information by the user.
- Minimize the memory load: Users should not need to remember information from previous screens.
- Data display should be compatible with data entry: Formats such as dates, should match the format data must be introduced.
- Provide flexible user control: In cases when the user requires, the information should be changeable by the user, such as providing alphabetical order in data results.

4.3 User's attention

For different aspects of interaction, a temporal change on visual display of information may help us to get the user's attention. This may benefit different aspects such as information introduction (getting the user's attention to the current field of interest). Some elements that can be used to get the attention of the users are:

- Intensity: A few levels of intensity should be used in forms, and high intensity can be used to emphasize a field.
- Marking: Underlining or using other elements such arrows, bullets, asterisks and so on.
- Size: The maximum number of sizes that should be used is four. Larger sizes attract more attention.
- Choice of fonts: A maximum of three fonts is advised.
- Inverse video.
- Blinking: Since it is a very attractive feature, this should be used with prudence.
- Color: At most 4 colors should be used, and additional colors can be reserved for occasional use.
- Audio: Soft tones can be used for regular positive feedback and harsh sounds for rare emergency conditions.

Overusing different elements may produce cluttered, uncomfortable interfaces. These elements should be used with care. Some side effects can be artificially relating unrelated items (matching them in intensity, colour or fonts will visually link them). Audio tones may be irritating to some users.

4.4 Facilitating data entry

Data-entry tasks may occupy a large portion of user's time, and can be an important source of frustration and potentially dangerous errors. Some aspects that must be taken into account:

- Design transactions consistently: Similar sequences of actions should be performed for similar tasks.
- Minimize the amount of inputs: The lower the number of inputs required for a task, the higher the productivity of the users. For instance, although menu selection may relieve the user to memorize different options, the need of change the hand's position to take the mouse, may cancel its advantage. Advanced or expert users may prefer to type six to eight characters instead to move to a menu, and it will require less amount of time.
- Reduce memory load: Lengthy lists of codes or complex command strings must be avoided.
- Data display and data entry must be consistent: As stated, this is an especially important issue when different formats may translate to very different information bits.
- Flexibility for user control of data entry: Some advanced users may prefer to enter the data in a certain order. This may collide with consistency, but sometimes user's comfort may him or her feel safer and this may be important for life-critical operations such as air-traffic control.

5. Universal Design Principles

There is a large amount of knowledge in different areas of design that may help us when designing user interfaces. In this section we will introduce some of the principles that have arose from many years of practice from different areas. Most of them come from the famous *Universal Principles of Design* book, from where we extracted the ones we found were more notable or useful. Some principles have already been visited, but will be illustrated here with a more detailed explanation or examples.

5.1 The 80/20 Rule

The 80/20 rule asserts that approximately 80 percentage of the effects generated by any large system are caused by only the 20 percentage of the variables in that system.

This rule is observed in all large systems, including economics (i. e. 80% of a company revenue comes from 20% of its products), computer systems (80% of errors are caused by 20% of the components), and so on. In our case, 80% of the usage in an application will be focused on only 20% of its features.

It is a useful rule for focusing resources, that is, we must focus most of our attention to the 20% of the features that are mostly used, to the 20% of the features that are critical, and so on. Focusing on aspects of the system that are beyond the critical 20% rapidly yields diminishing returns.

The rule must be used to assess the value of elements, and determine the target of our efforts in design and optimization.

5.2 Barrier-free design

Accessible or barrier-free designs have four characteristics:

- **Perceptibility:** Everyone can perceive the design, regardless of sensory abilities. This feature is achieved when everyone correctly understands the user interfaces. We must design interfaces with colour-blind friendly designs, with redundant visual indicators (such as text and colour), and so on.
- **Operability:** It indicates that anyone, regardless of physical abilities, is able to work with our system. In User Interface design, this means that we must take care of people with motor inabilities or slower response times (such as for double-click, as a typical case in elder people or children). This can be achieved by facilitating the access to controls (i. e. avoiding key combinations that require the use of two hands).
- **Simplicity:** It is achieved when everyone can understand the design regardless of experience, literacy, or concentration level. Unnecessary complexity must be removed. Designs must be clear, concise, and consistent, and feedback must be provided clearly.
- **Forgiveness:** It is achieved when the design minimizes the occurrence of errors and, in case of error, it also minimizes the errors' consequences. The use of good affordances, constraints, proper input formatting, and confirmations when required help to achieve forgiveness.

5.3 Aesthetic-Usability Effect

Aesthetics play an important role in the way designs are used. Aesthetic designs look easier to use, and encourage its use more than non aesthetic designs. This effect produces the perception that an aesthetic design is easier to use than a non-aesthetic design. However, you can find many examples that indicate the contrary. For example, the iPhone Calendar, though beautifully designed, makes a poor use of space, thus avoiding the visualization of a complete day in our calendar. On the other side, WebOS calendar is able to show a higher amount of information in a smaller space due to some elements such as an intelligent free-time collapsing option (see Figure 2). In use, the iPhone Calendar also requires a higher amount of clicks to generate an appointment, as compared to other calendars such as Android's or WebOS.

Figure 2: Left: iPhone Calendar. Right: Palm's WebOS calendar. WebOS Calendar makes a much smarter use of the space with the collapsing free-space regions and the overlapping of the view changing buttons. All in all, more things are shown in a 3.1-inch screen (right) than the famous 3.5-inch screen of the iPhone (left).

Since aesthetic designs are perceived as easier to use, we must devote important efforts to improving our designs. They are usually more readily accepted and user over time, and also foster positive relationships with people.

5.4 Correct alignment

We have already seen the infamous *butterfly ballot* from the US presidential election in 2000. One of the main problems was alignment.

Elements in a design must be aligned to each other. This creates a sense of unity and cohesion, as well as facilitates reading. When using grid or column-based alignments, we are also guiding the reading directions of the users. This is important because we may guide the attention, as well as induce the relation between the different elements.

Most common and useful alignments are rows and columns. Although more elaborate designs can exist, it may be necessary to add other visual cues to enforce the alignment direction.

5.5 Chunking

The term chunk is used to refer to a unit of information in short-term memory. Chunking is a technique that seeks to place the information in a way that accommodates to the limits the humans have to process bits of information. The famous paper by George Miller *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information*, studies the capabilities of human brain in recalling performance. Essentially, smaller chunks are easier to remember than larger lists, for example, most people can remember a list of 5 words for 30 seconds, but few can remember a list of ten words for 30 seconds.

Chunking is a technique that refers to elements that must be memorized, such as menu items, telephone numbers and so on. It is not required to divide all the elements in a screen or page in groups of 5 or so, since this can yield no benefits at all. Tasks such as text scanning do not require recalling the text that has been analysed; therefore, elements such as dictionary pages must not be chunked.

5.6 Cognitive dissonance

Cognitive dissonance is a psychological phenomenon that refers to the discomfort felt at a discrepancy between what you already know or believe, and new information or interpretation. It therefore occurs when there is a need to accommodate new ideas, and it may be necessary for it to develop so that we become "open" to them.

This discomfort is released usually in one of these three ways:

- Changing our behaviour.
- Justifying our behaviour by changing the conflicting cognition.
- Justifying our behaviour by adding new cognitions.

A typical example of cognitive dissonance is smoking. Many smokers feel some discomfort with the fact that smoking will shorten their lives and that they want to have long and healthy lives. There are different ways to reduce cognitive dissonance among smokers:

- Changing the behaviour: Quitting smoking.
- Justifying the behaviour by changing the conflicting cognition: Denying the evidence that smoking produces cancer.
- Justifying the behaviour by adding new cognitions: Rationalizing their behaviour by saying that only a few amount of hard smokers actually become ill.

A typical example of cognitive dissonance is a service that offers a free trial period but that requires a long time and effort to make the service properly work for you. When the free time expires, it will be a hard work to change the system and therefore, the larger the

cognitive dissonance that will lead to a large amount of people to accept the price to continue with the service. Cloud services integrated into computers or mobile phones may produce a high cognitive dissonance if we want to change our system with a different one, since moving all our music, folders, configurations, and so on, may be time costly, and therefore we might prefer to buy a new product from the same company. The larger we are involved (the higher number of services we use), the higher the cognitive dissonance to change the trademark.

5.7 Colour

Colour is an important feature that can make a design more visually pleasing and aesthetic. They can be used to reinforce layout design and the meaning of elements. Some aspects must be taken into account when using colours.

- **Number of colours:** The number of colours must be kept low. Commonly, up to five is enough. Do not rely on the colour to provide information, always use a second cue because there are users with some sort of colour-blindness.
- **Colour combinations:** When using different colours for adjacent elements, we may take different approaches. The most common, and widely accepted are the use of analogous colours (adjacent colours on the colour wheel), complementary colours (opposing colours on the colour wheel), colours at the corners of a symmetrical polygon circumscribed in the colour wheel, triadic (triangle) or quadratic (square), or combinations of colours found in nature (see Figure 3).
- **Saturation:** Attention attraction can be achieved using saturated colours (pure hues), when performance and efficiency are important, the use of desaturated colours may help. Desaturated colours are perceived as more professional, while saturated colours are perceived as more exciting and dynamic.
- **Symbolism:** The meanings of colours may vary among cultures, therefore, one should not try to use a certain colour to produce some feeling (i. e. dark colours are assumed to make people sleepy) without a previous verification of the target audience.

Figure 3: Colour combinations found in nature: The right image shows a portion of Irish coast, and some of its colours have been picked for the Dublin website.

5.8 Consistency

Consistency is a well-known design principle. It may be applied or classified in different ways:

- **Internal consistency:** It refers to how the elements of the application are consistent with each other. It induces trust.
- **External consistency:** It refers to the consistency of the elements of a design with other elements in the environments or other applications. It extends the advantages of internal consistency across multiple systems.
- **Aesthetic consistency:** It refers to the consistency of the style and appearance. The use of logos or trademarks with the same font and colours makes the elements more recognizable than when using different fonts and/or colour combinations.
- **Functional consistency:** It refers to the coherency between expected results and effective results of our actions. If our interface is functionally consistent, the learning curve will be softer, and usability will be improved.

5.9 Five hat racks

Five hat racks refers to the ways of organizing information. There are five ways to organize information: category (similarity relatedness), time (chronological sequence), location (geographical or spatial references), alphabet (alphabetical sequence), and continuum (magnitude; highest to lowest, best to worse).

The organization of information is a powerful way to improve understanding as well as influencing the way people think. The LATCH principle (Location, Alphabet, Time, Category, Hierarchy) is a slight redefinition of the *Five Hat Racks*. The five organization modes are described next.

Location

Location is chosen when the information you are comparing comes from several different sources or locales. Doctors use different locations of the body to group and study medicine. Concerning an industry you might want to know where on the world goods are distributed.

Alphabet

Alphabet is best used when you have enormous amount of data. For example words in a dictionary or names in a telephone. As usually everybody is familiar with the Alphabet, categorizing by Alphabet is recommendable when not all the audience is familiar with different kind of groupings or categories you could use instead.

Time

Time is the best form of categorization for events that happen over fixed durations. Meeting schedules or our calendar are examples. The work of important persons might be displayed as timeline as well. Time is an easily framework in which changes can be observed and comparisons made.

Category

Category is an organization type often used for goods and industries. Shops and services in the yellow pages are easy to find by category. Retail stores that sell clothing have separated parts for men, woman, and children clothing. This mode works well to organizing items of similar importance.

Hierarchy

Hierarchy organizes by magnitude. From small to large, least expensive to most expensive, by order of importance, etc. Hierarchy is to be used if you want to assign weight or value to the ordered information.

5.10 Garbage In – Garbage Out

This principle refers to the long-time experience of computer scientists that have found out that good input produces good output while bad input often produces bad output. It is often abbreviated GIGO. This metaphor refers to two common input problems:

- **Problem of type:** This problem appears when the incorrect type of input is provided to a system. Sometimes the type problem can be detected, but may produce the high level of garbage if it goes undetected. Elements such as numerical fields that can be fed with a phone number or a credit card number are examples of this type problem.
- **Problem of quality:** It occurs when the correct type of input is fed into a system but it has some defects. This type of errors may often be caused accidentally, and one way to minimize problems is the use of previews and confirmations.

The best way to avoid garbage out is to properly avoid garbage in: Type checks, input formatting, default values, or example inputs may help to reduce the input of garbage data.

5.11 Iconic Representation

It is the use of images to represent objects, actions and concepts. There are four types of iconic representations:

- **Similar:** They try to represent the action or object by an image that is visually similar to the element they try to represent. It is often useful for simple objects or actions (right turn) but less effective when trying to represent complex concepts.
- **Example:** They use elements that can be related with the object or action that they represent, such as a plane to indicate an airport.
- **Symbolic:** Images have a high level of abstraction. They are more effective when the symbol is well-established (such as the unlock icon with an open lock).
- **Arbitrary:** They are icons that use images with no relationship with the element or action, such as the male/female symbols.

5.12 Law of Prägnanz

The word *prägnanz* is a German term meaning "good figure." The law of Prägnanz is sometimes referred to as the law of good figure or the law of simplicity. It is referred to as one of the Gestalt Principles of perception.

This law holds that objects in the environment are seen in a way that makes them appear as simple as possible.

We tend to order our experience in a manner that is regular, orderly, symmetric, and simple. Some of the Gestalt Laws relevant for visual design are:

- **The law of closure:** The mind may experience elements it does not perceive through sensation, in order to complete a regular figure (that is, to increase regularity).
- **The law of similarity:** The mind groups similar elements into collective entities or totalities. This similarity might depend on relationships of form, colour, size, or brightness.
- **The law of proximity:** Spatial or temporal proximity of elements may induce the mind to perceive a collective or totality.
- **The law of symmetry:** Symmetrical images are perceived collectively, even in spite of distance.
- **The law of continuity:** The mind continues visual, auditory, and kinetic patterns.
- **The law of common fate:** Elements with the same moving direction are perceived as a collective or unit.

Some examples of these laws are shown in Figure 4.

Figure 4: Examples of the Gestalt principles of similarity (left), proximity (centre) and closure (right).

5.13 Occam's Razor

Occam's Razor (also Ockham's Razor) is sometimes expressed in Latin as *lex parsimoniae* (the law of parsimony, economy or succinctness). It is a principle that generally recommends selecting from among competing hypotheses the one that makes the fewest new assumptions. In design it is applied to indicate that it is preferred the simplicity over the complexity.

Occam's original statement was "Entities should not be multiplied without necessity". Albert Einstein's formulation was: "Everything should be made as simple as possible, but not simpler".

Unnecessary elements decrease design's efficiency. If we compare Yahoo's search page with Google's search page, we should prefer the second to the former. However, we should not forget that Yahoo's webpage also provides other services than search.

5.14 Orientation Sensitivity

The efficiency with which we can perceive the orientation of lines is limited. Humans easily distinguish or can judge vertical or horizontal orientations while oblique orientations are more difficult to distinguish. Usually 30 degrees is the minimum recommended difference in orientation for the users to perceive it properly. This is due to two main phenomena in visual perception:

- **Oblique effect:** The relative deficiency in perceptual performance for oblique contours as compared to the performance for horizontal or vertical contours. It is caused by a greater sensitivity of neurons to vertical and horizontal stimuli than to oblique stimuli.
- **Pop-out effect:** It is the tendency of certain elements in a display to pop out as figure elements, and therefore be easily detectable. For instance, in a set of lines, targets are more easily detectable if they differ a minimum of 30 degrees over the other background of lines. When combined with the oblique effect, it becomes stronger: it is easier to distinguish a line with a subtle difference in orientation if it is close to a vertical or a horizontal line rather than close to a set of oblique lines.

5.15 Pictorial superiority effect

Concepts are much more likely to be remembered experientially if they are presented as pictures rather than as words. In many cases, information recall is superior when the information is presented in pictures. However, this happens after thirty seconds, that is, when the information is recalled before 30 seconds, the same amount of information can be recalled in text than in pictures. However, after 30 seconds, it is easier to recall pictorial information.

This also happens when the time of exposure is small, images can be better recalled than text.

5.16 Progressive Disclosure

It is an interaction design technique that sequences information and actions across several screens in order to reduce feelings of overwhelm for the user. It keeps displays clean and uncluttered. By disclosing information progressively, you reveal only the essentials and help the user manage the complexity of feature-rich sites or applications to reduce confusion, frustration, or disorientation. Progressive disclosure is not just about displaying abstract then specific information, but rather about getting the user's attention by going from simple to more complex actions.

Sometimes, designers present too much information in order to reduce kinematic load. Since not all the elements in an interface will be equally used, progressive disclosure will move complex and less frequently used options out of the main user interface and into secondary screens.

5.17 Rule of thirds

The rule of thirds is a compositional rule of thumb in visual arts such as painting, photography and design. The basic principle behind the rule of thirds is to imagine breaking an image down into thirds (both horizontally and vertically) so that you have 9 parts. With this grid in mind the *rule of thirds* identifies four important parts of the image that you should consider placing points of interest in as you frame your image: the two central vertical edges, and the two horizontal edges of the grid. Then, important compositional elements should be placed along these lines or their intersections.

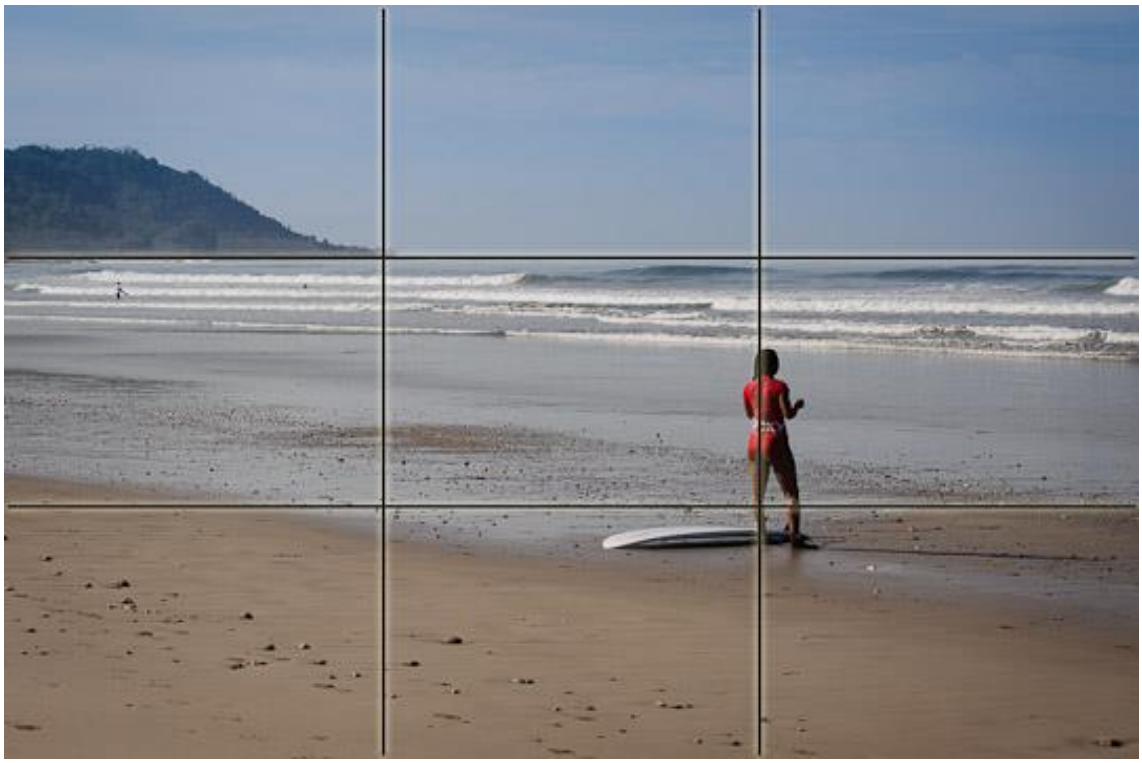


Figure 5: Example image following the rule of thirds.

5.18 Signal to noise ratio

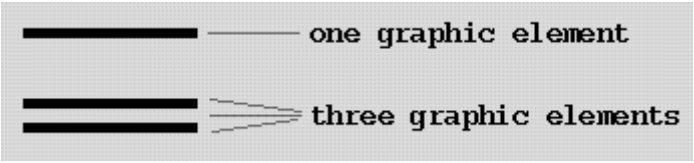
Signal-to-noise ratio is a measure used in science and engineering that compares the level of a desired signal to the level of background noise. It is defined as the ratio of signal power to the noise power. A ratio higher than 1:1 indicates more signal than noise. The goal of communication is maximizing signal and minimizing noise.

We can improve the signal to noise ratio in our designs by keeping them simple. We can enhance information by using redundant coding and highlighting, and we can remove noise by eliminating unnecessary elements. For tabular data, for example, lines may sometimes be removed if proper alignment or other background solid colours are used to enhance alignments. Bar charts, for instance must avoid textures.

5.19 1+1 = 3

Since all the elements in a design are perceived together, some of them, although not designed initially this way, interact. This may create non-information patterns and texture.

This effect is properly illustrated here:



The single line is clearly a single graphic element, while when we add a second line, the empty space lying in between creates a third visual element although it was not intended to. As a consequence, we have three graphical elements, each of the lines and *the space between them*.

This effect is called 1 + 1 = 3. The general result of this effect is that there is an extra effort of the visual system processing something that is not information, but noise and clutter. The noise produced by the 1+1=3 effect is directly proportional to the contrast in value (light/dark) between figure and ground. Therefore, different contrast colors might alleviate the incidental clutter.

A particularly common example is caused by the use of boxes around text, which are commonly discouraged.

6. Usability testing

6.1 Introduction

Usability testing has as primary goal to provide guidance to product developers for the purpose of improving the ease-of-use of their products.

For an introduction to usability testing and how to prepare a usability test, you may refer to pages 1-26 of the following Technical Report:

Lewis, J. R. (2005). *Usability testing* (Tech. Report 29.3820). Raleigh, NC: IBM Corp.: http://drjim.0catch.com/usabilitytesting-ral.pdf.

6.2. How many users are enough?

6.2.1 Introduction

A common mistake is thinking that usability testing is very complex and costly and therefore should be reserved for large design projects. Large usability tests are possible. However, many authors claim that a low number of users (i. e. from 3 to 10) may be enough for solving the most important problems of a design.

The safest answer to this question is: 'It depends'. Actually, the problem is not properly defined. When performing usability testing, we may address different issues:

- Reveal all the usability issues on an application

- Identify most important problems
- Do an academic study
- Inform a complete site redesign
- Persuade somebody in the organization that there is a major problem with the site or application

Of course, other variables must be taken into account such as the budget and time to perform the study, the diversity of target audience of the application or site, or the strategic importance of the application or site. As a consequence, we will always face a trade off between objectives, time and money, user variability, and strategic importance. Although at first sight one may think that the usability study has as main goal the identification of all usability problems, this is not correct. An important issue to take into account is the fact that running a usability test is not the solution, but only part of it: After problems have been identified, they must be solved, and often there may be not enough budget to correct them all.

Moreover, one will never know if all problems have been detected: a new tester may find a previously unearthed problem. Finally, there is another element to consider: when fixing problems, one may also introduce a new one. Typically, one must balance between the importance of the corrected problem and the relative importance of the new problem that may be introduced.

6.2.2 Importance of the usability problems

As already stated, we should first define what we are going to solve. Commonly, in medium and large organizations, the problem to solve may be stated as: "How many testers do we need to find the 'big' usability issues that stop users having a good experience?" When evaluating such problems, they must be classified by its importance. Usually, we can label the problems as being into any of these three groups:

1. Usability catastrophes: They prevent users from achieving their goals
2. Serious usability problems: Problems that interfere, but do not prevent goal achievement
3. Cosmetic usability problems: They may irritate users some degree, but may not interfere with achieving their goal

Most organizations are concerned with the first and second usability problems. Thus, they must be detected and solved. Furthermore, solving the third class will improve the user experience.

Many researchers found that the 20/80 rule also applies for usability studies: the 20% of the problems account for the 80% of the bad user experience. So finding the large problems and fixing them will definitely improve user experience a great deal.

6.2.3 Again, how many users?

If we stick to the original question, we can find different answers in literature.

[Virzi, 1992] found that 80% of known usability problems could be surfaced with 5 testers, and 3 that testers would reveal the most severe problems. [Nielsen & Landauer, 1993] state that the best benefits are usually obtained testing no more than 5 users and running as many small tests you can afford. They also find a way to model the number of usability problems found in a usability test:

$$N(1-(1-L)^n)$$

Where *N* is the total number of usability problems in the design and *L* is the proportion of usability problems discovered while testing a single user. They also say that 70% of these problems can be revealed using 3 testers. However, as said, we cannot predetermine the number of existing problems. Moreover, other issues affect this model, such as the quality of the tests performed. As found by [Faulkner, 2003], the variability of the testers can have enormous impact on the number of usability issues discovered. In some studies, she found that different groups of 5 testers revealed a number of 'known' usability problems that ranges from 55% to 99%. The main conclusion is that the test users must be representative of the target population. It is then an important, even critical task, to define this target population. This is by no means trivial, and in many cases may be a highly complex problem to address.

[Nielsen, 2000] indicates that there is a law of diminishing returns – "the third tester will do many things that you have already observed with the first or second user… [and] generate a small amount of new data… after the fifth user you are wasting your time by observing the same findings repeatedly but not learning much new". Although they also suggest that 15 users will reveal all the known usability problems in a design, the common recommendation is to use 5 users with three iterative tests. Therefore, the approach would be to make a first test with 5 users. This would reveal roughly the 80% of the application or site problems. After fixing them, a second round of tests with the same users may unearth new problems or remaining ones. Moreover, this second test is useful to assess issues such as information architecture, task flow, or user needs. Finally, a third test should be enough. The experiments have shown that an iterative test with 5 users yields better results than a single test with 15 testers.

[Perfetti & Landesman, 2001] suggest that more than 8 testers are needed to detect all usability issues. But they do not suggest doing these all at once; rather, they advocate a programme of ongoing iterative testing "bringing in a user or two every week". [Molich, 2006] also states that "the number of users needed for web-testing depends on the goal of the test. If you have no goal, then anything (including nothing) will do". He suggests 3-4 users to *'sell' usability* into the organisation, 5-6 users to find *catastrophic problems* in an iterative process, and over 50 users to find all usability problems in an interface.

6.2.4 References

[Faulkner, 2003] Faulkner, L. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. Behavior Research Methods, Instruments & Computers, 35, 3, Psychonomic Society (2003), 379--383.

[Kruger, 2006] Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition, 2006.

[Nielsen, 2000] Nielsen, J., Why you only need to test with 5 users, www.useit.com/alertbox/20000319.html, Mar, 2000.

[Nielsen & Landauer, 1993] Jakob Nielsen and Thomas K. Landauer. 1993. A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems* (CHI '93). ACM, New York, NY, USA, 206-213.

[Molich, 2006] R. Molich, 'Usability Testing Best Practices: An Interview with Rolf Molich' – UI 11 2006 Conference Articles (www.uie.com)

[Perfetti, C. & Landesman, L., 2001], C. Perfetti and L. Landesman, Eight is not enough, http://www.uie.com/articles/eight_is_not_enough/, Jun, 2001.

[Virzi, 1992] Robert A. Virzi. 1992. Refining the test phase of usability evaluation: how many subjects is enough? *Hum. Factors* 34, 4 (August 1992), 457-468.

7. Model-View-Controller

7.1 Introduction

Model-View-Controller is a programming pattern that decouples data, program logic, and display. The pattern was first proposed by Trygve Reenskaug, a Smalltalk developer at the Xerox Palo Alto Research Center, in 1979. Although its use is very extended, many programmers have used it without being aware of its name and history. Many applications that require a GUI are designed using the Model-View-Controller architecture.

Model-view-controller has three important parts:

- Model: The data we use. In a Computer Graphics application, it is usually the geometric model of the scene.
- View: Its task is to render the contents of a model. In a Computer Graphics application, it consists of the implementation of the methods required to display the information on screen: OpenGL code and Qt display methods (resizeGL, paintGL, and so on) form this part.
- Controller: It is the part of the code that controls the interaction and acts on the model and the display. In a Computer Graphics application, it is usually all the code that manages mouse and key inputs.

7.2 Model View Controller in a Graphics application

There are different approaches for the Model-View-Controller architectures. The Apple Cocoa framework proposes a model slightly different from the original SmallTalk's approach, probably more suitable for a graphics application. The different elements of the Model-View-Controller design interact with each other with method invocations and events, as shown in Figure 6.

The difference between this proposal and the original one is that the notifications of state changes in model objects are communicated to the view through the controller. Hence the controller mediates the flow of data between model and view objects in both directions. View objects, as always, use the controller to translate user actions into property updates

on the model. In addition, changes in model state are communicated to view objects through an application's controller objects.

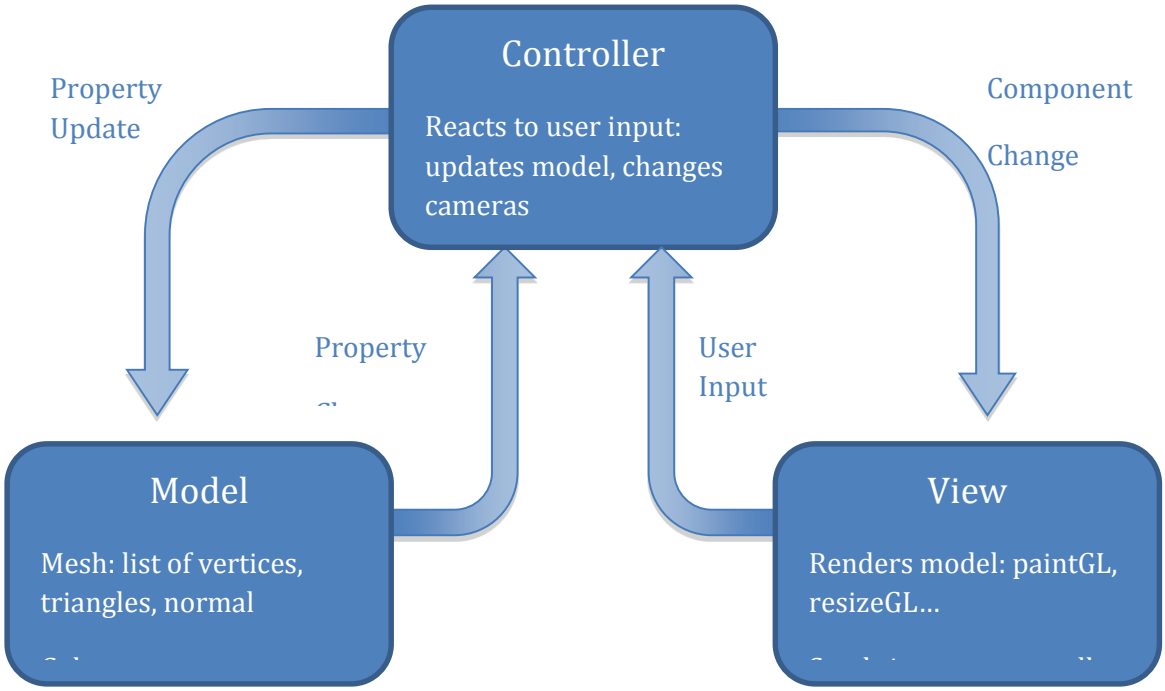


Figure 6: The Model View Controller in a Graphics application.

8. Further Reading

8.1 Compulsory readings

The following articles are compulsory. You must read them, since they are complementary to these course notes.

- Miller, G.A., The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information, The Psychological Review, 1956, vol. 63, pp. 81-97, http://www.psych.utoronto.ca/~peterson/psy430s2001/Miller%20GA%20Magical%20Seven%20Psych%20Review%201955.pdf
- Nielsen, J., First Rule of Usability? Don't Listen to Users, Jakob Nielsen's Alertbox, August 5, 2001: http://www.nngroup.com/articles/first-rule-of-usability-dont-listen-to-users/
- Nielsen, J., Optimizing a Screen for Mobile Use, Jakob Nielsen's Alertbox, March 28, 2011: http://www.nngroup.com/articles/optimizing-a-screen-for-mobile-use/

- Nielsen, J., Mobile Content Is Twice as Difficult, Jakob Nielsen's Alertbox, February 28, 2011: http://www.nngroup.com/articles/mobile-content-is-twice-as-difficult/
- Nielsen, J. Mental Models, Jakob Nielsen's Alertbox, October 18, 2010: http://www.nngroup.com/articles/mental-models/
- Nielsen, J., Scrolling and Attention, Jakob Nielsen's Alertbox, March 22, 2010: http://www.nngroup.com/articles/scrolling-and-attention/

8.2 Extra readings

We believe those readings can also be useful, although they are not compulsory:

- Nielsen, J., Horizontal Attention Leans Left, Jakob Nielsen's Alertbox, April 6, 2010: http://www.nngroup.com/articles/horizontal-attention-leans-left/
- Nielsen, J. Kinect Gestural UI: First Impressions, Jakob Nielsen's Alertbox, December 27, 2010: http://www.nngroup.com/articles/kinect-gestural-ui-first-impressions/
- Michael Zuschlag, Achieving and Balancing Consistency in User Interface Design, http://www.uxmatters.com/mt/archives/2010/07/achieving-and-balancing-consistency-in-user-interface-design.php
- Nielsen, J. Fresh vs Familiar. How aggressively redesign?, http://www.nngroup.com/articles/fresh-vs-familiar-aggressive-redesign/
- Chen, J. The impact of Aesthetics on Attitudes Towards Websites, http://www.usability.gov/articles/062009news.html