

Usability

Mike Jackson
Software Architect

EPSRC

NERC SCIENCE OF THE ENVIRONMENT

 **archer**

CRAY
THE SUPERCOMPUTER COMPANY

epcc



Usability looks at developing products and tools that are both useful and easy to use. In this lecture, I'll give an introduction to usability and why it is important. I'll also describe a set of heuristics for the design of software that is usable. These heuristics, or rules for good design, do not just apply to graphical user interfaces or web sites, but to batch tools, command-line tools, configuration files, and to many objects in the real world too.

Reusing this material



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_US

This means you are free to copy and redistribute the material and adapt and build on the material under the following terms: You must give appropriate credit, provide a link to the license and indicate if changes were made. If you adapt or build on the material you must distribute your work under the same license as the original.

Note that this presentation contains images owned by others. Please seek their permission before reusing these images.



Why bother?



Image by U.S Pacific Fleet



In July 1988, Iran Air Flight 655 was shot down by the USS Vincennes in the Persian Gulf. There have been many explanations as to why this happened, including political, psychological and technical.

The usability of the ship's radar systems has been singled out for criticism.

Three large displays showed every contact.

However, to get speed, range, and altitude, an operator had to explicitly punch up that information, which was displayed on a tiny monitor.

This itself is a problem, but the system was also missing data.

Rates of change are important in assessing whether an approaching aircraft is a threat.

One that is accelerating and descending is a greater threat than one that is ascending.

Crew members had to compare data taken at different times and make the calculation in their heads, on scratch pads, or on a calculator, and in a highly-tense environment.

The contractor who developed the display focused not on good design but on cost and giving the Navy what said it needed, not what it actually needed.

See "Overwhelmed by Technology: How did user interface failures on board the USS Vincennes lead to 290 dead?" Luke Swartz, 2001.

<http://xenon.stanford.edu/~lswartz/vincennes.pdf>

Image: <https://www.flickr.com/photos/compacflt/5098411676/in/photolist-8LwGaA-9h989F-osmGVt-ooz2mE-9MTQ8e-9KehR2-bhvMWM-5EnKSv-cfVGHj-deJjmy-6ija7d-6ijaEw-p4y8kf-pkLUy2-p4yH8J-p4xWe6-d1YUSS-d1tkrL-pYgSr6-pYgSrg-bhvN1g-8LwGhf-9MTPqx-9MWzTd-9MTQbK-9MTPzM-9Kh7Hm-9Kh88s-9Kh75J-9Kh7rA-8LtCpa-deJjds-deJjiV-cfVFt7-cfVH97-deJiBS-cfVGVW-cfVGsy-deJis3-cfVFJu-cfVFZW-deJjzx-6adcRH-9NFVNX-9NJybd-9NGKrb-9NKJWL-9NH11D-9NEwpU-9NGabs>

Desire paths



Image by webnetwork



Usability problems often occur due to a mismatch between designers, or, in our case, software developers, and users.

Designers and developers give the users what they think they want rather than what users actually want.

An example of this mismatch can often be found in public parks, where grassy areas often have dirt tracks worn across them.

These tracks are called desire paths or social trails.

The public choose the most efficient way through the park from A to B, which may not be where the park's planner has chosen to put their paths, possibly for reasons of artistic appeal.

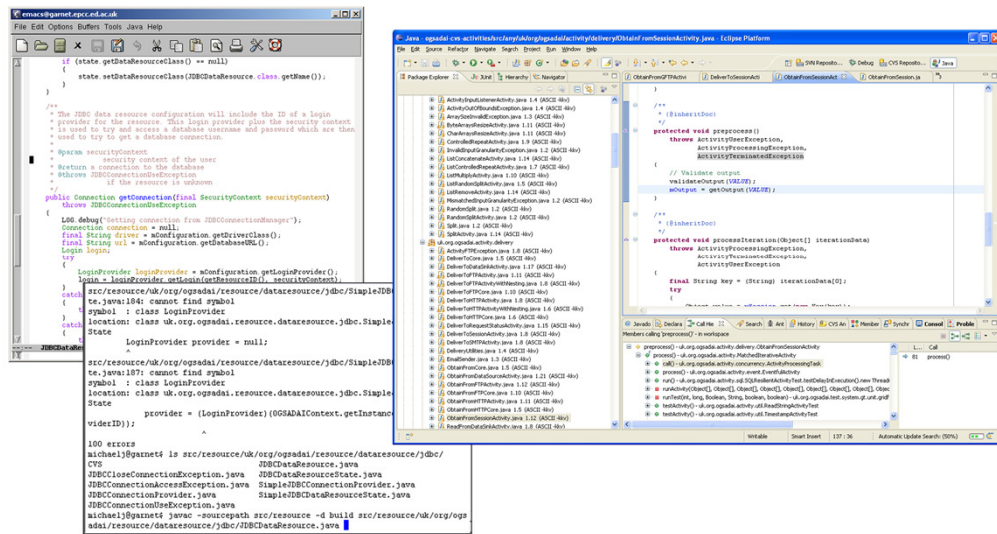
There is an apocryphal story of a university that did not lay any paths in their new campus during its first year.

After the first year had passed, they built the paths over the desire paths, an example of user-centred design.

See "Desire paths", Jussi Ahola, 2 December, 2013, <https://medium.com/design-ux/2b6b5b0f0e92>

Image: <https://www.flickr.com/photos/wetwebwork/2847766967/>

Which is the most usable?



Consider the task of writing code.

Which is more usable?

A text editor and a command-line compiler and debugger or an integrated development environment such as Eclipse or Visual C++?

Neither or both!

It depends on the user, their skills, knowledge, background, education, experiences to date and personal preferences.

Heuristics for good usability

- “practical method not guaranteed to be optimal or perfect”, Wikipedia
- Nielsen, J., and Molich, R. (1990) Heuristic evaluation of user interfaces, Proc. ACM CHI'90 Conf. Seattle, WA, 1-5 April 1990, pp249-256. DOI: 10.1145/97243.97281
- Factor analysis of 249 usability problems
- Nielsen, J. (1994). Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods, John Wiley & Sons, New York, NY, pp25-62.
- 10 Usability Heuristics for User Interface Design
- J. Nielsen, 01/01/1995, <http://www.nngroup.com/articles/ten-usability-heuristics/>



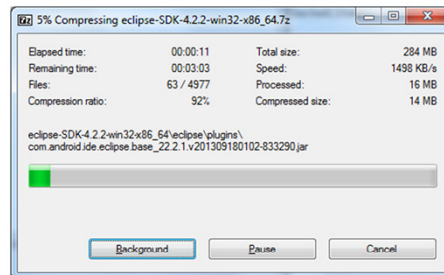
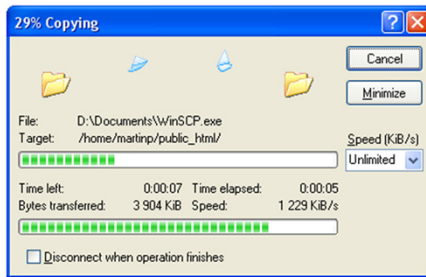
Wikipedia defines a heuristic as a “practical method not guaranteed to be optimal or perfect”.

Jakob Nielsen and Rolf Molich proposed a set of heuristics for usability, as part of an expert approach to evaluating the usability of a user interface, called a heuristic evaluation.

Nielsen then analysed 249 usability problems and, from these, proposed 10 heuristics for user interface design.

These heuristics can be used for both the design of usable products or tools and also to assess how usable a product or tool is, to identify where the usability problems lie, and to propose how its usability can be improved.

1. Visibility of system status



```
$ python fslinstaller.py
--- FSL Installer - Version 2.0.12 ---
[Warning] Some operations of the installer require administrative rights, for example installing
into the default folder of /usr/local. If your account is an 'Administrator' (you have 'sudo'
rights) then you will be prompted for your administrator password when necessary.
[OK] Installer is current
Where would you like to install FSL? [/usr/local]: /disk/ssi-dev0/home/michaelj/fsl
Install location doesn't exist, should I create it? [yes]: yes
[sudo] password for mjj:
Downloading FSL version 5.0.6 (this may take some time)
57131008/1583996239 - 3%
```



“Visibility of system status” is defined as: “The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.”

If not observed, there is a risk that the user may think the software has frozen or run into other problems and might exit prematurely.

They may waste time, or lose work, as a result.

Appropriate feedback can include progress bars and status information presented as part of a graphical user interface or progress information displayed at the command-line.

2. Match between system and the real world

'sin(Pi / 2)';

$\sin\left(\frac{1}{2} \pi\right)$

Static analysis of a truss: Method of joints

$L_1=2000$, $L_2=1000$, $L_3=2000$

$-F_{AB} - F_{AD} \cos(\theta) = 0$ (0.1)
 $-2000[N] + F_{AD} \sin(\theta) = 0$ (0.2)
 $-2000[N] + F_{AD} \sin(\theta) = 0$ (0.3)

Solve for unknown forces:

$\text{eval}(\text{solve}(\text{sin}(\theta) = \frac{1000}{2000}, \cos(\theta) = \frac{1000}{2000}))$

$40000[N] \sin(\theta) - 4[N] F_B = 0, R_D = 0, -3000[N] + F_B + R_D = 0, -F_{AB} - \frac{1}{2} F_{AD} = 0, -2000[N] + \frac{1}{2} F_{AD} = 0, F_{AB} - F_{BC}$ (4.1)
 $-\frac{1}{2} F_{AD} + \frac{1}{2} F_{DE} = 0, -1000[N] + \frac{1}{2} F_{AB} + \frac{1}{2} F_{BC} = 0, R_C = 0, R_D = 0, \frac{1}{2} F_{AD} = 0, \frac{1}{2} F_{DE} = 0, -F_{DE} - \frac{1}{2} F_{BC}$
 $+\frac{1}{2} F_{AD} = 0, -\frac{1}{2} F_{DE} - \frac{1}{2} F_{BC} = 0, F_{DE} + \frac{1}{2} F_{BC} = 0, -\frac{1}{2} F_{BC} = 0, F_B - \frac{1}{2} F_{AD} = 0, \frac{1}{2} F_{AD} = 0$ (4.2)

$R_D = -2000[N], F_{AB} = -1500[N], F_{BC} = 3750[N], F_{DE} = 3000[N], F_{AD} = -2500[N], F_B = 10000[N], F_{CD} = 3750[N], F_{DE}$
 $= 2500[N], F_{BC} = -5250[N], N = 0, R_C = 0$

3750 [N] $\text{symbol table} = 343.0335366 [10^3]$

“Click a circle widget” versus “Select a molecule”



“Match between system and real world” is defined as “The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.”

An interface that displays mathematical symbols and formulae, as they are written on the page, speaks the language of science more directly than one which renders these into textual equivalents, for example “sin pi slash two”.

In a chemistry package, asking a user to “click the circle widget” is speaking the language of a software developer, whereas “select a molecule” is speaking the language of a chemist.

2. Match between system and the real world



In the 1980s, characters in maze games on home computers were controlled with the cursor or “arrow” keys on the keyboard.

The motivation behind this design is that a user can look at keyboard for reminder of what key makes your character move in what direction.

On the ZX Spectrum, these keys were 5, 6, 7 and 8 on the top row of the keyboard.

For moving a character up and down, this required using the 6 key to go down and 7 to go up.

So, to move a character up and down required pressing keys that were adjacent to each other horizontally.

Designers listened to the complaints of users and, instead, defined key combinations such QAOP or AZNM for up, down, left and right movement.

So, to move a character up and down required pressing keys QA or AZ that were above and below each other.

Users were less inclined to make mistakes, their character would live longer, and they could play their games more easily.

This is termed minimising the gap between “intention” – what a user wants to do – and “action” – what the user actually has to do to carry out their intention.

2. Match between system and the real world



Image by Angus Lepper






Fast Ticket train ticket machines seem to have just enough depth between their touch-sensitive glass and the actual display that, unless you are of an average height, the buttons appear to be slightly offset from where they actually are.

There are a number of hardware solutions to this problem but one software solution could be to shrink the visible size of the buttons while increasing their margin for error in the touch-screen sensors.

From MSc High Performance Computing student, Angus Lepper, 2015.

2. Match between system and the real world

The image shows three overlapping windows. The top-left window is the 'DICOM Confidential Policy Editor' with a menu bar (File, Resources, Security, Policy, Help) and various toolbars. The top-right window is an 'Input' dialog box with a question mark icon, a text field containing 'www.medical.com', and 'OK' and 'Cancel' buttons. The bottom window is a command prompt titled 'C:\Windows\system32\cmd.exe' displaying a Java stack trace. The stack trace starts with 'Caused by: java.lang.IllegalArgumentException: URI is not absolute' and lists several frames from the 'org.jdesktop.application' and 'uk.ac.sinsapse.policyeditor' packages, ending with 'at java.awt.AWTEventMulticaster.mouseReleased<Unknown Source>'.

This heuristic also tries to ensure that information only understandable to developers, such as implementation details, does not leak into the user interface.

For example, this medical package expects a URL to be provided.

If an invalid URL is provided then a Java stack trace is displayed.

Would a medical researcher understand this Java stack trace?

If a user ever sees an exception message or code fragment anywhere in a user interface then a developer has not done their job properly!

3. User control and freedom



“User control and freedom” is defined as “Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.”

This includes:

- Allowing users to cancel tasks that are taking too long, such as copying large files.
- Asking them to confirm actions that may take too long or may have security implications, such as running an executable downloaded via a web browser.
- Supporting undo and redo.
- Asking for confirmation of irreversible actions, such as deleting files.

As an example, on Linux/Unix systems, running “rm” to remove a file will often immediately remove the file whereas deleting a file within Windows puts it in the recycle bin, so it can be retrieved if it was accidentally deleted.

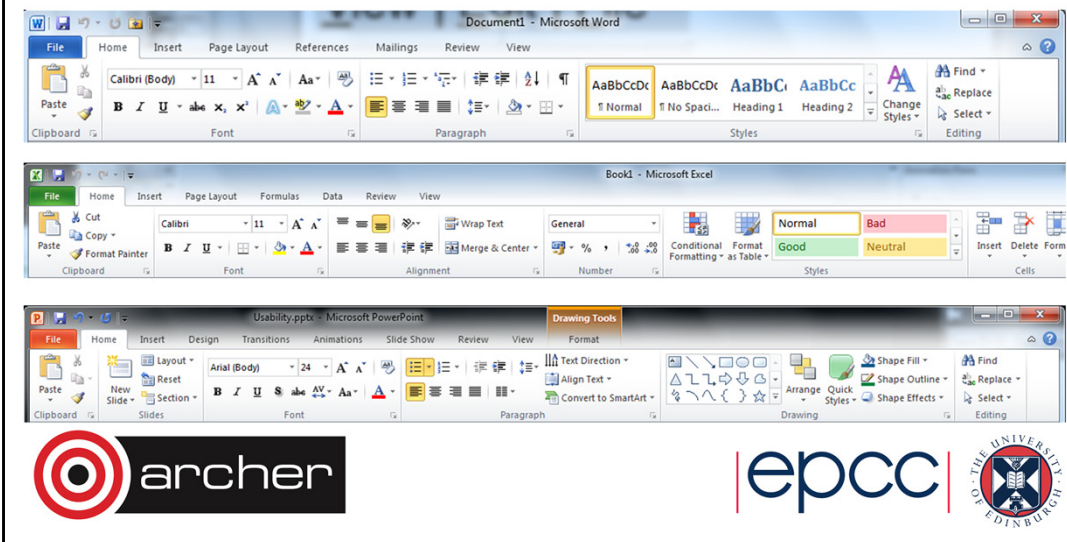
3. User control and freedom



Amazon's account menu is problematic in terms of user control and freedom. If I want to log out, the only option is "Not Michael? Sign Out" but I am Michael! Do I sign out there too? This is not a clearly marked exit!

4. Consistency and standards

View | Edit | File File | Edit | View



“Consistency and standards” is defined as “Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.”

A menu bar that presents from left-to-right a View menu, then an Edit menu, then a File menu is less usable than one that presents from left-to-right a File menu, then an Edit menu, then a View menu.

This is because the latter design is far more consistent with myriad products and tools than the former.

A similar comment applies to placing the Help menu at the far right-hand-side of a menu bar.

Consistency helps users learn to use, and remember how to use, products and tools as it allows them to apply their prior knowledge.

Product suites often adopt a consistent look-and-feel across products to allow knowledge of how to use one product to be transferred to other products in the suite. For example, Microsoft Word, Excel and PowerPoint all share many common menus, tabs and icons.

4. Consistency and standards

```
$ git --version  
git version 1.8.4
```

```
$ make --version  
GNU Make 3.81  
Copyright (C) 2006 Free Software  
Foundation, Inc.  
This is free software; see the source  
for copying conditions.  
There is NO warranty; not even for  
MERCHANTABILITY or FITNESS FOR A  
PARTICULAR PURPOSE.
```

```
This program built for x86_64-redhat-  
linux-gnu
```

```
$ python --version  
Python 2.6.6
```

```
$ ls --help
```

```
Usage: ls [OPTION]... [FILE]...  
List information about the FILES (the  
current directory by default).  
Sort entries alphabetically if none of -  
cftuvSUX nor --sort.
```

```
$ du --help
```

```
Usage: du [OPTION]... [FILE]...  
or: du [OPTION]... --files0-from=F  
Summarize disk usage of each FILE,  
recursively for directories.
```

```
$ python --help
```

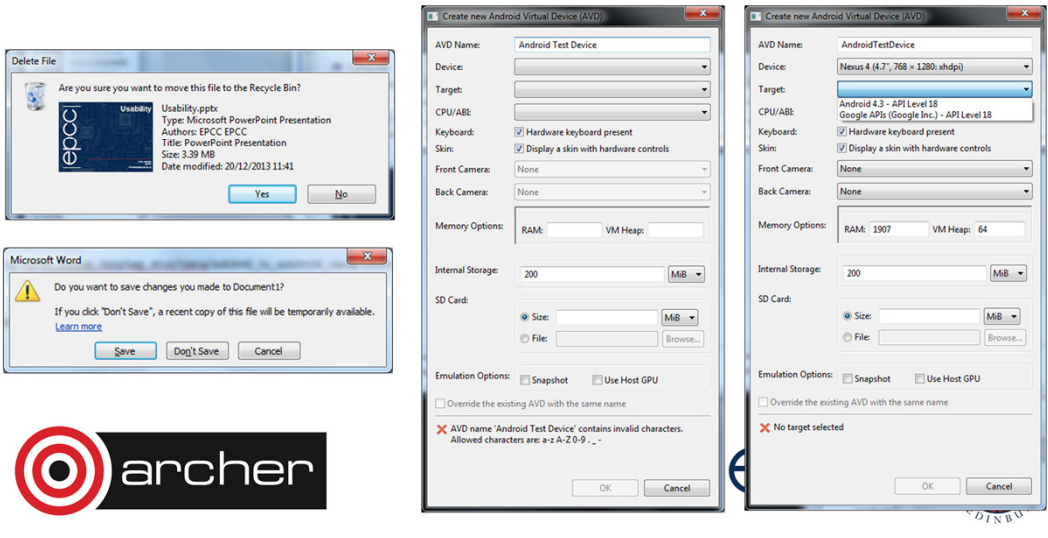
```
usage: python [option] ... [-c cmd | -m  
mod | file | -] [arg] ...
```



Many Linux/UNIX tools support a set of common command-line flags.
For example `--version` for version information and `--help` for help information.

5. Error prevention

```
$ rm SSI-QuBic-pid.docx
rm: remove regular file `SSI-QuBic-pid.docx'?
```



“Error prevention” is defined as “Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.”

Linux/UNIX can be configured so that users are asked to confirm whether they want to delete a file when they use the “rm” command, to prevent them from accidentally deleting a file.

Windows, likewise, asks users to confirm file deletions, before moving files to the recycle bin.

Many products and tools remind users if they have unsaved work if they select to exit, and give them the opportunity of saving this work, so they don’t accidentally lose their work.

Some tools configure their user interfaces to ensure that only valid options can be selected.

For example, after selecting a type of device in the Android Toolkit, only valid target APIs can be selected by the user.

This prevents the user from creating a virtual device with an invalid configuration.

5. Error prevention



"... the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used...Affordances provide strong clues to the operations of things. Plates are for pushing. Knobs are for turning. Slots are for inserting things into. Balls are for throwing or bouncing. When affordances are taken advantage of, the user knows what to do just by looking: no picture, label, or instruction needed."

Likewise, bars on doors are for pulling.

The James Clark Maxwell building had a door with both a sign that said "pull" and a vertical metal bar.

The other side of the door had both a sign that said "push" and a vertical metal bar. This was a poor design since someone might push the door if:

1. They didn't read the sign.
2. They did read the sign but their subconscious didn't deem it important enough to pass on to the conscious part of their brain, if they were in a hurry.
3. They do not know English.
4. They could not see, either because they were blind or there was dense smoke in the corridor, for example.

The problem could be solved, if, on the "push" side of the door, instead of a vertical metal bar, a vertical metal plate was placed.

A metal plate affords pushing.

The door also violates "Match between system and the real world" and, more importantly, "consistency and standards".

In some buildings, fire doors have flat panels on the “push” side and routes to fire exits are through a sequence of doors to be pushed open.

So long as you are seeing, or feeling, doors with flat panels ahead, you know you are heading to the exit.

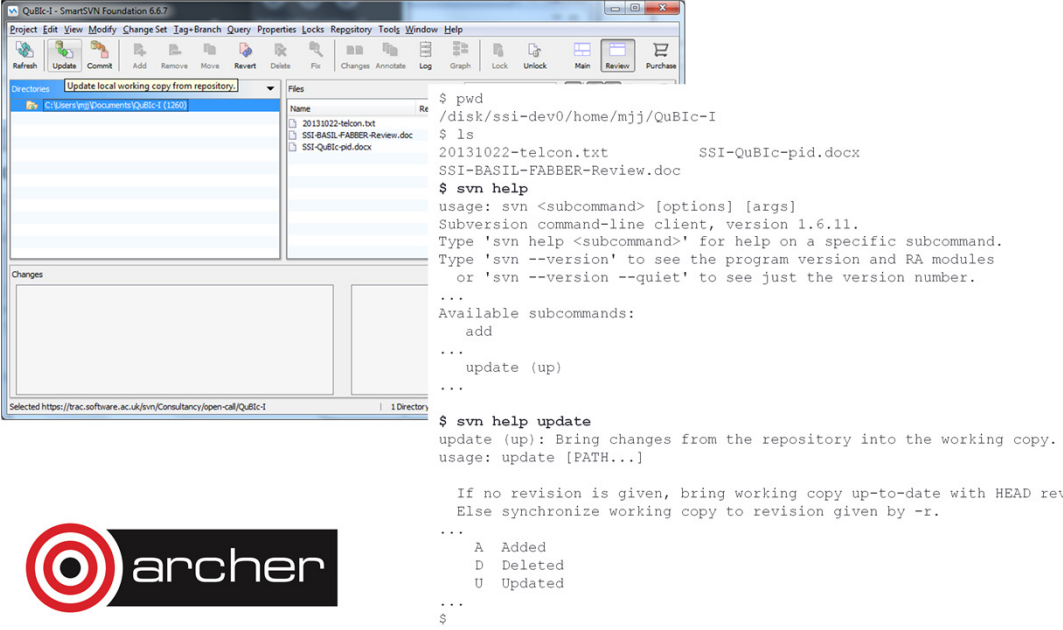
Another problem is that if the door was opened to its full width then your knuckles were grazed.

This could be prevented by putting in a door-stop.

In this case there was a door-stop but it was there only to prevent the handle from banging into the wall, not your knuckles!

From MSc High Performance Computing, student Marton Feigl, 2015.

6. Recognition rather than recall



The screenshot shows the SmartSVN Foundation 6.7 application window. The interface includes a menu bar (Project, Edit, View, Modify, Change Set, Tag+Branch, Query, Properties, Locks, Repository, Tools, Window, Help) and a toolbar with icons for Refresh, Update, Commit, Add, Remove, Move, Revert, Delete, File, Changes, Annotate, Log, Graph, Lock, Unlock, Main, Review, and Purchase. The main area is divided into several panes: a 'Directories' pane on the left showing the current path 'C:\Users\mjj\Documents\QuBic-I (1200)', a 'Files' pane in the center showing a list of files with columns for Name and Re, and a 'Changes' pane at the bottom left. The 'Files' pane contains the following entries:

Name	Re
20131022-telcon.txt	
SSI-BASIL-FABBER-Review.doc	
SSI-QuBic-pid.docx	


The terminal window on the right shows the following output:

```
$ pwd
/disk/ssi-dev0/home/mjj/QuBic-I
$ ls
20131022-telcon.txt          SSI-QuBic-pid.docx
SSI-BASIL-FABBER-Review.doc

$ svn help
usage: svn <subcommand> [options] [args]
Subversion command-line client, version 1.6.11.
Type 'svn help <subcommand>' for help on a specific subcommand.
Type 'svn --version' to see the program version and RA modules
or 'svn --version --quiet' to see just the version number.
...
Available subcommands:
  add
  ...
  update (up)
  ...

$ svn help update
update (up): Bring changes from the repository into the working copy.
usage: update [PATH...]

    If no revision is given, bring working copy up-to-date with HEAD rev
    Else synchronize working copy to revision given by -r.
...
  A Added
  D Deleted
  U Updated
...
$
```



“Recognition rather than recall” is defined as “Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.”

Graphical user interfaces promote recognition via menus, icons, buttons and tool-tips. A user can recognise the parts of the interface, from their previous experience with this or other tools, and so decide what actions they need to take to carry out their task. Command-line interfaces require recall as a user has to remember what commands to run, and how.

Alternatively, they have to Google!

Reducing what a user has to remember can improve usability.

Bash, FTP, GNUplot, and many Linux/UNIX tools, support built-in help, both about the tools as a whole, and specific commands, so all the user needs to remember is the command to get help.

Many command-line tools automatically display help if no arguments to the command are provided.

7. Flexibility and efficiency of use

The image shows a composite screenshot illustrating keyboard shortcuts and terminal history. On the left, a Microsoft PowerPoint window is shown with the 'File' menu open. A red arrow points to the 'File' menu item, labeled 'ALT shortcut'. Another red arrow points to the 'Find...' option in the 'Edit' menu, labeled 'CTRL shortcut'. Below the PowerPoint window is the 'archer' logo. On the right, a terminal window displays the output of the 'history' command, showing a list of previously executed commands and their line numbers. Below the terminal window is the 'epcc' logo and the University of Edinburgh logo.

```
$ history
494 ./go.sh
495 pine
496 xterm
497 pine
498 timex515
499 xtimest&
500 ./go.sh
501 ./go.sh
502 ls
503 pine
504 history
$ !500
```

“Flexibility and efficiency of use” is defined as “Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.”

Using the keyboard has been proven by research to be faster than using the mouse. Allowing users to press TAB to move between buttons and fields in a graphical user-interface can lead to faster use than if they have to select fields and press buttons using the mouse.

This is especially true if the user has to carry out a repeated sequence of actions.

Likewise CTRL and ALT shortcuts allow users to invoke commands within menus.

Linux/UNIX provides keyboard shortcuts to allow previous commands that have been run to be browsed, edited and re-run.

The “history” command also allows previously-executed commands to be viewed and rerun.

Returning to the train ticket machine example, certain machines do not allow combinations of railcard and non-railcard purchases in a single transaction.

This means that the user has to make two separate transactions.

This could be improved by allowing users to combine these purchases within a single transaction.

7. Flexibility and efficiency of use



Image by jaygooby



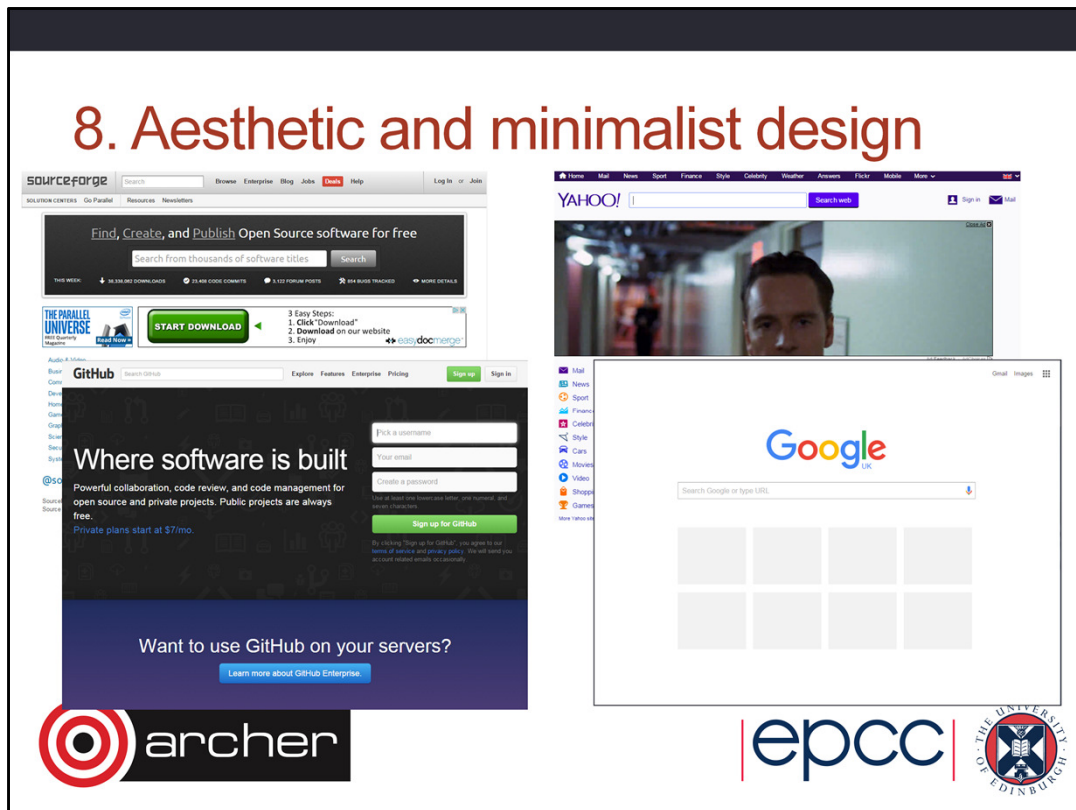
Some self-checkout machines allow you to put your own bag on the scale. But some require a staff member to acknowledge that you have done this. Instead of waiting for a staff member, customers often unload a precarious pile onto the scale and then pack their bag only after they pay. The customer adopts a non-simple and natural “dialog”, for more flexible and efficient use.

It would be easier for customers if staff did not have to verify bags

Image: <https://www.flickr.com/photos/jaygooby/2055347396>

From MSc High Performance Computing, student Angus Lepper, 2015.

8. Aesthetic and minimalist design



“Aesthetic and minimalist design” is defined as “Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.”

SourceForge has projects of the month and editors picks, compared to GitHub which just has a sign up form.

Yahoo’s front page is cluttered with all sorts of content.

Google’s is stripped down to the bare essentials, a search form.

9. Help users recognise, diagnose, and recover from errors

```
$ filter configfile.xml  
java.lang.NullPointerException
```

```
$ filter configfile.xml  
Error
```

```
$ filter configFile.xml  
File error
```

```
$ filter configFile.xml  
XML parsing error at line 4 character 2 in  
configFile.xml. Unknown element: databaseUL
```



“Help users recognise, diagnose, and recover from errors” is defined as “Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.”

Imagine a program to apply some filter to a file.

A `NullPointerException` indicates an error, but nothing else.

A message, “Error”, at least avoids exposing developer-specific details to the user, but is not much more useful.

A message, “File error”, tells the user that the problem is with their file, giving them a clue as to where the problem lies.

A message “XML parsing error at line 4 character 2 in configFile.xml. Unknown element: databaseUL”, tells the user exactly what the error is and where it arises and a clue as to how it can be fixed, by sorting the unknown “databaseUL” element.

9. Help users recognise, diagnose, and recover from errors but be discreet

EASE Invalid Input

Provided login appears to be invalid

Username:

Password:

EASE Authentication Required

Password incorrect.

- Check if [caps lock] is on?
- Forgotten your password? Use the relevant link from the Help box to proceed.

GitHub Search or type a command

Incorrect username or password.

amazon [Your Account](#) | [Help](#)

There was a problem with your request
There was an error with your E-Mail/Password combination. Please try again.

This rule needs to be applied more subtly for interfaces that support authentication and authorisation.

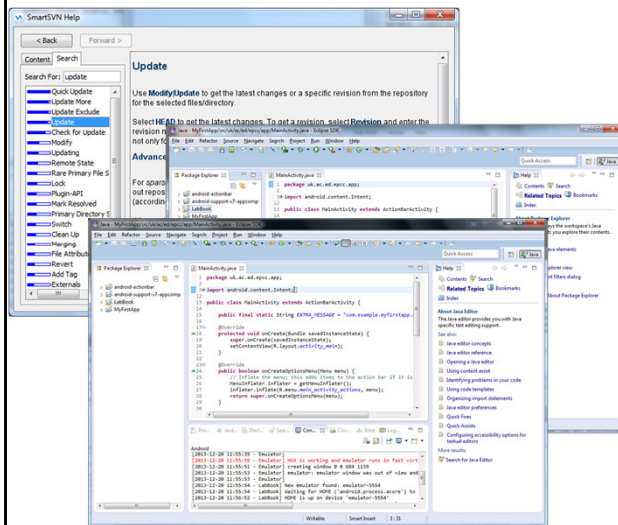
The EASE user interface lets a malicious user know whether the username or password was incorrect.

So, they could find out what the valid user names are.

GitHub and Amazon are more subtle and more security-conscious.

They just tell a user when their login has failed but not specifically whether it was the username or password that was wrong.

10. Help and documentation



```
$ svn --help
usage: svn <subcommand> [options]
Subversion command-line client, ve
Type 'svn help <subcommand>' for h
subcommand.
```

```
$ svn help update
update (up): Bring changes from th
working copy.
usage: update [PATH...]
```

```
$ man dot
DOT(1)
...
SYNOPSIS
dot [- (G|N|E)name=value] [
[-outfile] [-Klayout] [-
[files]
neato [- (G|N|E)name=value]
[-n[1|2]] [-outfile] [-Kla
[-V] [files]
```



“Help and documentation” is defined as “Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.”

Many products and tools, like SmartSVN, provide a Help menu that provides access to built-in help pages, which are searchable.

Eclipse has dynamic help and provide help specific to the part of the interface the user is working with, for example the Java package explorer or the Java text editor.

As described earlier, many Linux/UNIX shell commands support a “-h” or “--help” flag.

Some tools, that support a rich set of commands, for example Subversion or Git, provide command-specific help also.

Many Linux/UNIX commands also have searchable manual, or “man”, pages.

10. Help and documentation

What it does

≠

How it works

≠

How to use (build and run) it



Describing what a car does, is different to describing how a car works.

This, in turn, is different to describing how to drive.

It is the same for software.

Describing what software does is not the same as describing how it works.

This, in turn, is not the same as describing how to build and run it.

For research software, there is often a lot of information on what it does and how it does it, since this is where the research innovation is embodied.

However, the documentation on how actually to use it is sometimes neglected.

Providing a 10 minute quick start guide on how someone can use your software will help them to use it far more than a 100 page manual describing what your software does.

Likewise, explaining how they can use it on their data, to solve their problems, can help them even more, applying your software to their research, providing you with a means to demonstrate impact.

10. Help and documentation

- When you say...?
 - Python
 - Linux
 - C++ compiler
- Do you mean...?
 - Python 2 or Python 3
 - Scientific Linux 7 or Ubuntu
 - Cray, Intel or GNU C++, all three
- Prevent errors
- Show don't tell
 - "Run the filter command on your file"
 - "Run `$ filter configFile.xml`"



Good documentation also benefits from good design.

For example, if documentation says that a program needs Python, runs under Linux and has a library that compiles under C++, does this mean Python 2 and/or 3, Scientific Linux 7 and/or Ubuntu, the Cray, Intel, GNU C++ compilers or all three?

Being precise in terms of documentation helps prevent errors.

Likewise, instead of telling users what commands to run in paragraphs of English text, show them the commands to run, exactly as they should run them at the command line, ensuring a match between the system and its documentation.

“Very foolish words”

“Our software is usable because it has a graphical user interface”

“Our software has a user-friendly interface”



A graphical user interface does not guarantee a usable product or tool.

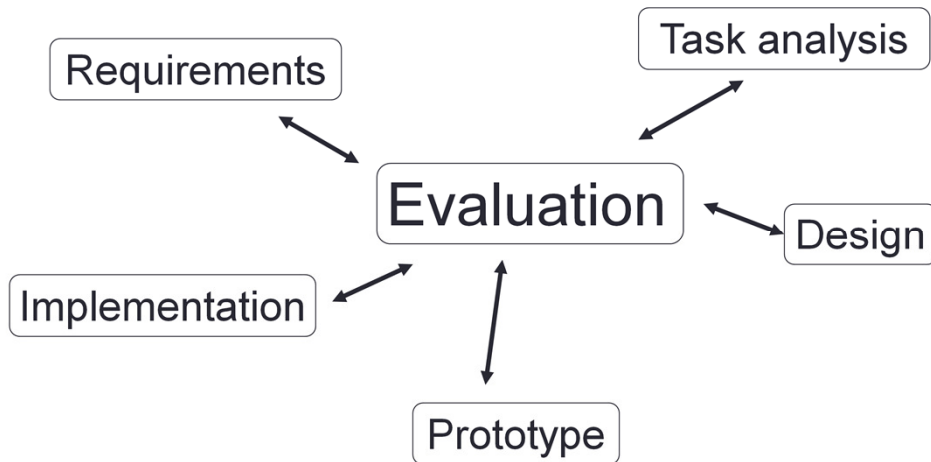
A badly designed graphical user interface can cause users far more problems than a well designed command-line interface.

Usability is a factor of the user, their task, technology (both hardware and software), and physical, social and organisational environment.

Don't say your software has a user-friendly interface because you are biased, you wrote it, of course it is user-friendly, to you!

By all means if your users say its user-friendly then quote them!

When do we consider usability?



Adapted from Hix, D. & Hartson, H. R. (1993). Developing user interfaces: Ensuring usability through product and process. New York, NY: John Wiley & Sons.



The star life cycle, from Hix and Hartson, shows that usability evaluation, assessing and considering issues of usability, contributes to all phases of software development whether this be gathering requirements, understanding the tasks users will do, designing the software, prototyping and implementation.

The star life cycle complements existing development models e.g. waterfall, rapid prototyping or agile methods to remind developers that usability should be considered always and everywhere.