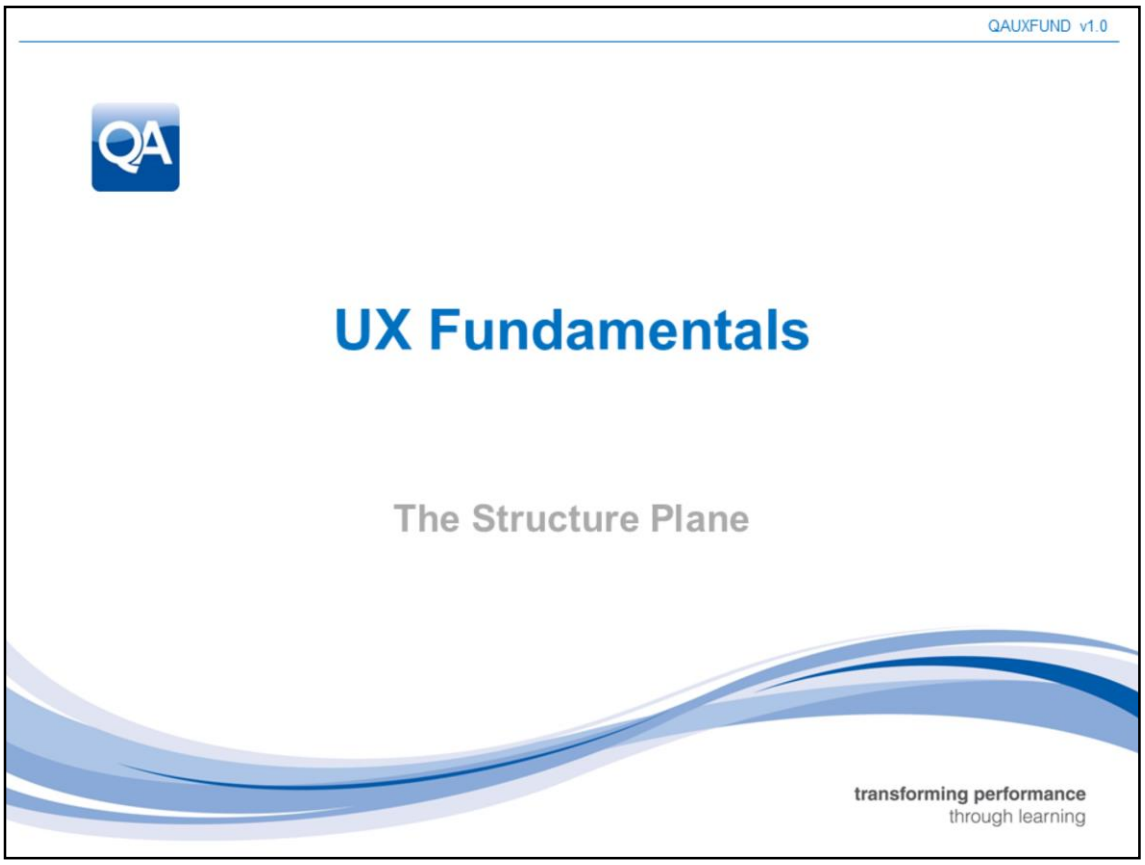

QAUXFUND v1.0
UX Fundamentals
The Structure Plane
transforming performance
through learning

QAUXFUND v1.0

The Structure Plane

- **The structure plane defines the way in which features fit together**
  - It is a shift from the more abstract information gathering phases
  - Moving towards concrete functionality

- **Also known as interaction design and information architecture**
  - We consider the organisation, grouping and categorising of content
  - Aiming to define patterns and sequences to presented to users
    - It deals with the way we convey information to users

2

QAUXFUND v1.0

Deliverables

- Information Architecture
- Conceptual Models
- Conventions
- Error Handling
- Language
- Workflow Diagrams

3

Navigation

- **Navigation Patterns are established IA concepts**
  - These patterns allow us to effectively categorise user interaction
  - Following established mental models for behaviour

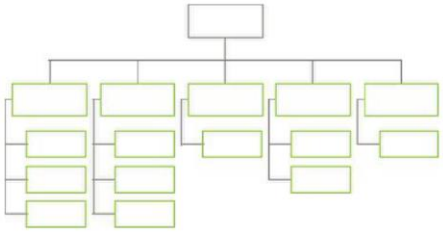
- **These include:**
  - Hierarchy
  - Database
  - Hypertext
  - Linear
  - Combined
  - Catalogue
  - Hub & Spoke

4

QAUXFUND v1.0

Hierarchical navigation

- **Hierarchies can be broad or deep**
  - Broad hierarchies have lots of top level items but few levels
  - Deep hierarchies have fewer top level but more sub levels



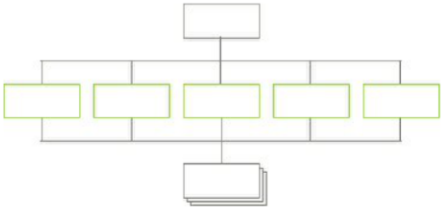
- **A hierarchy can also be described as strict or polyhierarchy:**
  - In a strict hierarchy an item can be in one and only one place.
  - In a polyhierarchy an item can be in more than one place.

5

QAUXFUND v1.0

Database

- **The database pattern is used for content with consistent structure**
  - The individual content usually have no connection to each other



- **The database pattern uses consistent fields and metadata**
  - Creating a page on an identical template

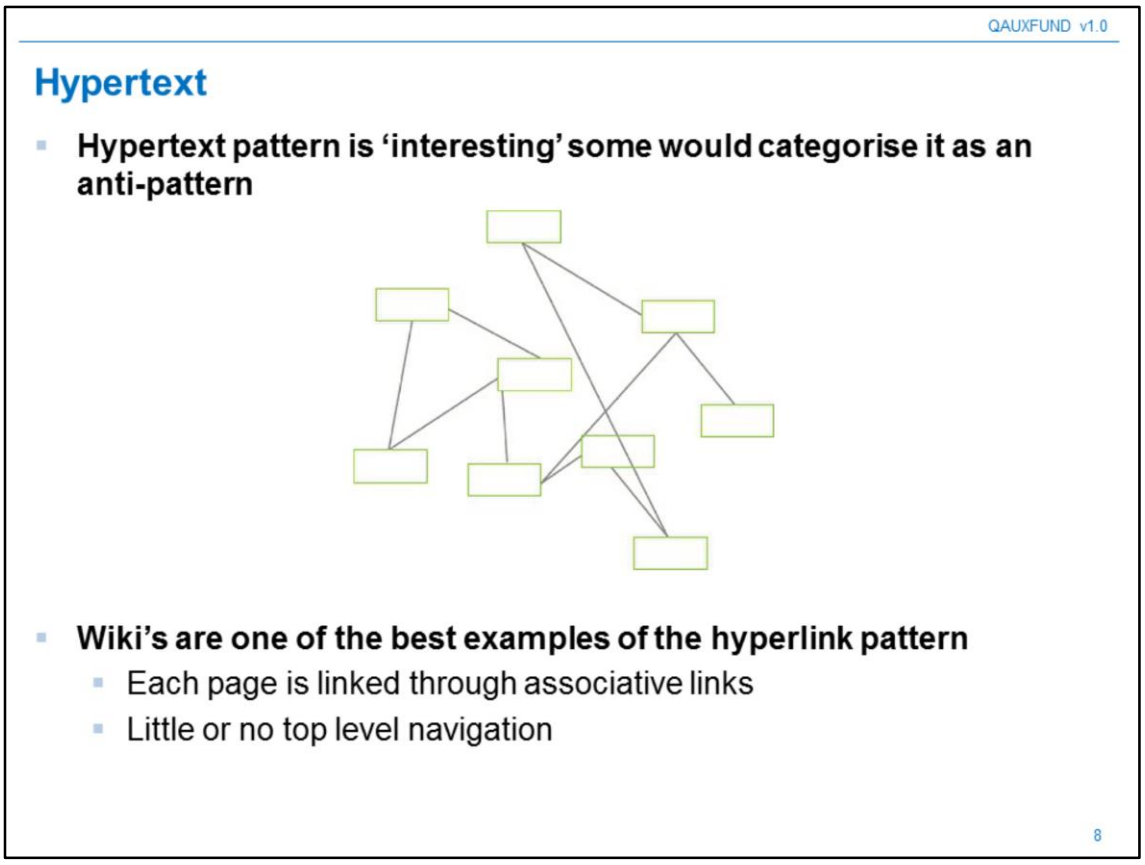
6

QAUXFUND v1.0

Database (2)

Consider the following two images – identify the similarities:

title description tags materials
location payment methods date added
photograph category colour



Hypertext structures are particularly useful when the content is being developed over a period of time and you don't know exactly what you are going to create.

In this situation it would be practically impossible to identify a detailed structure up front, or even to identify a basic pattern for a site. A lot of documentation projects start like this – people write individual pages of documentation as they get around to them and make relationships between them with links. Many sites that start with a hypertext structure are later re-organised, when the content is known.

Spencer, Donna (2011-04-13). A Practical Guide to Information Architecture (Kindle Locations 2318-2322). Five Simple Steps. Kindle Edition.

Linear

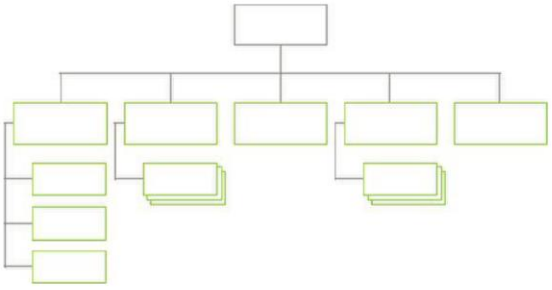
- **Linear navigation structures follow a straight line**
  - Rare in the web – checkout tasks are the most common

9

QAUXFUND v1.0

Combined Pattern

- **Most websites are a combination of multiple techniques**
  - Hierarchy & Database combined form the majority of websites



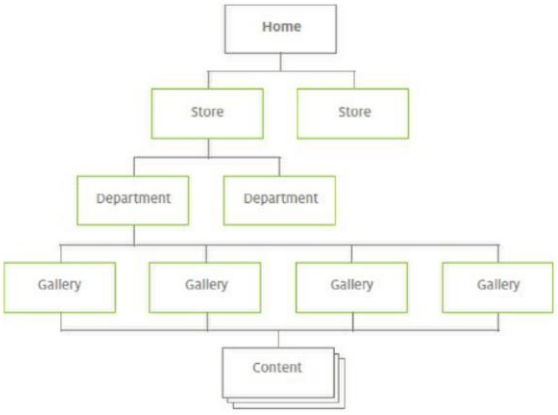
- **The challenge with this pattern is deciding what will be structured and what is data driven**

10

QAUXFUND v1.0

Catalogue

- **The catalogue pattern is a variant of the database pattern**
  - Particularly common in e-commerce.
    - At the bottom level is the content
    - Above that are up to three levels of hierarchy
    - Depending on the size of the site and type of content



11

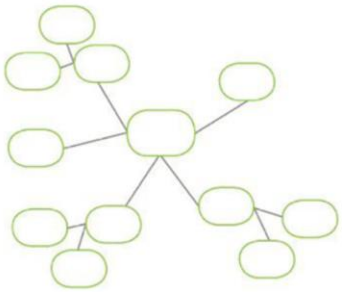
Jarred Spool examined this pattern and categorised that most pages were:

- Gallery pages: these provide direct access to the content pages
- Department pages: provide access to the galleries
- Store pages: provide access to the department pages

QAUXFUND v1.0

Hub & Spoke

- **Hub and spoke is an alternative hierarchical pattern**
  - Its difference comes from user usage

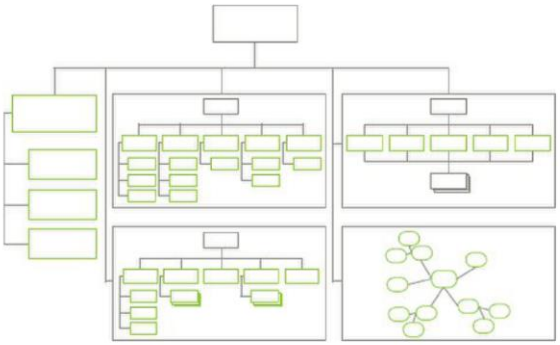


- **With a hierarchy, people tend to start at a point in it**
  - Then move down into deeper and deeper content
  - Often sticking within the one branch of the hierarchy.
- **With a hub and spoke, people move down one level**
  - Then return to the starting point (the hub)

12

Subsites

- **Large sites are often a combination of all of the above**
  - Do not fear using multiple IA metaphors if the UX data supports it



13

QAUXFUND v1.0

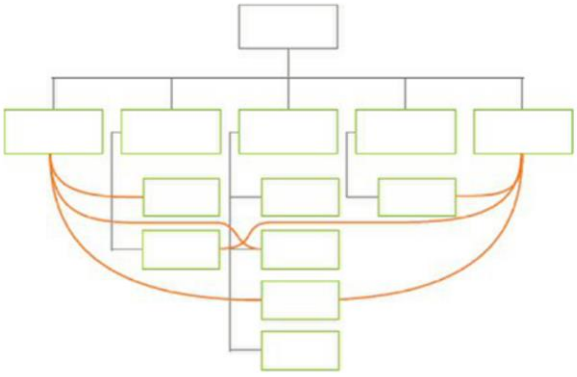
Focused Entry Points (1)

- **In many situations we will not be able to use a 'pure' pattern**
  - We need to modify the pattern based upon user mental models
  - Focused Entry points help us to achieve this

- **Users have different information needs and different experience**
  - In this situation focused entry points provide access points for tasks
  - Set up the site using the pattern that best suits the content
  - Matched with user need

14

Focused Entry Points (2)



15

QAUXFUND v1.0

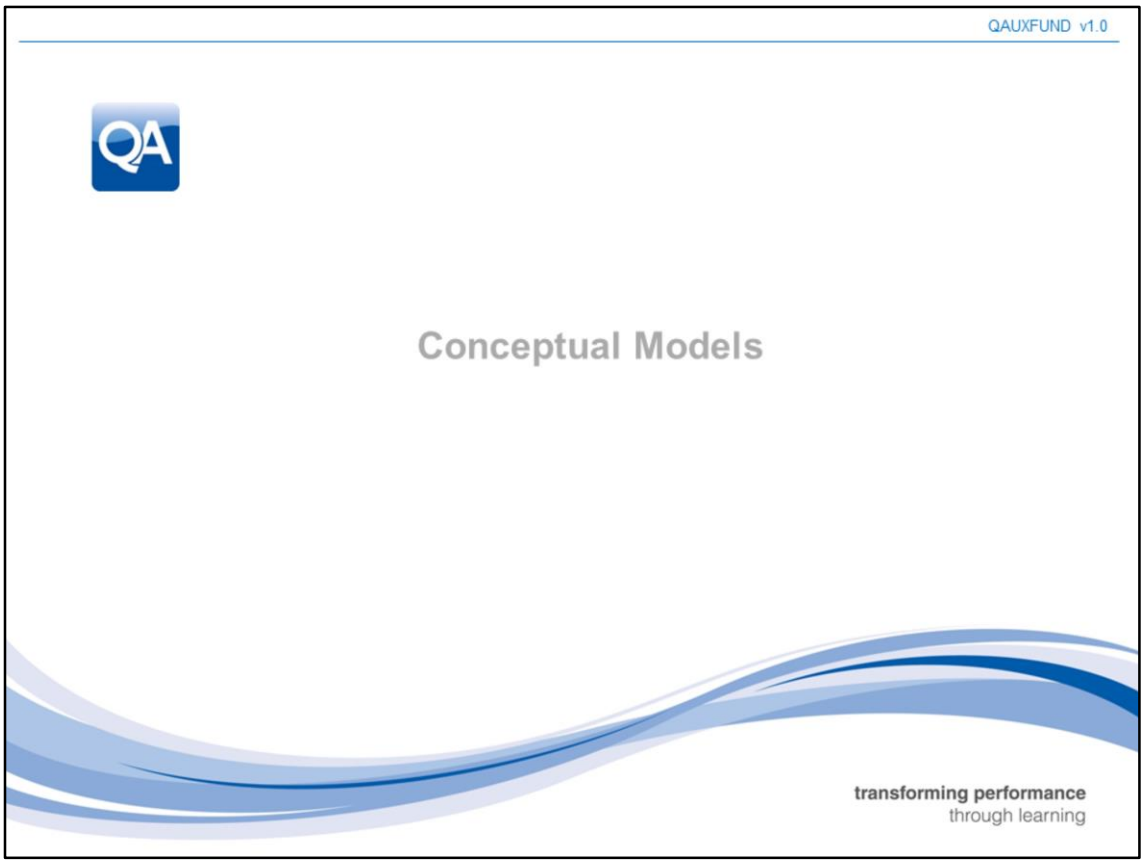
How to build an Information Architecture

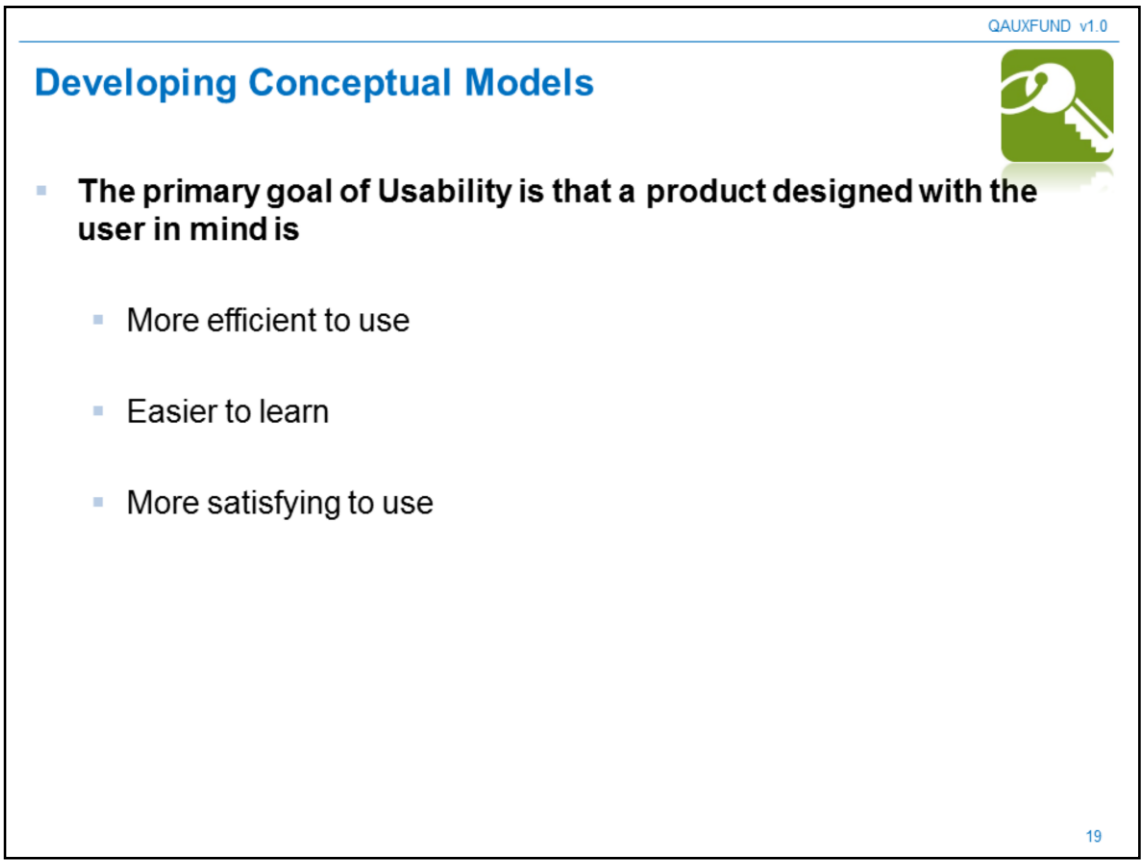
1. **Decide what we are creating**
   - Top level? Hierarchical?
   - If a database come up with the meta tags
2. **Just do it!**
   - Build it, if you are unsure try different patterns
   - Test with the user!
3. **Check it**
   - Verify your models with users, check against personas
   - With databases check the attributes and constraints
4. **Revise**
   - Go back to step 2!
5. **Stop**
   - Stop going back to step 2!
6. **Discuss**

16

Group Discussion – Navigation Models

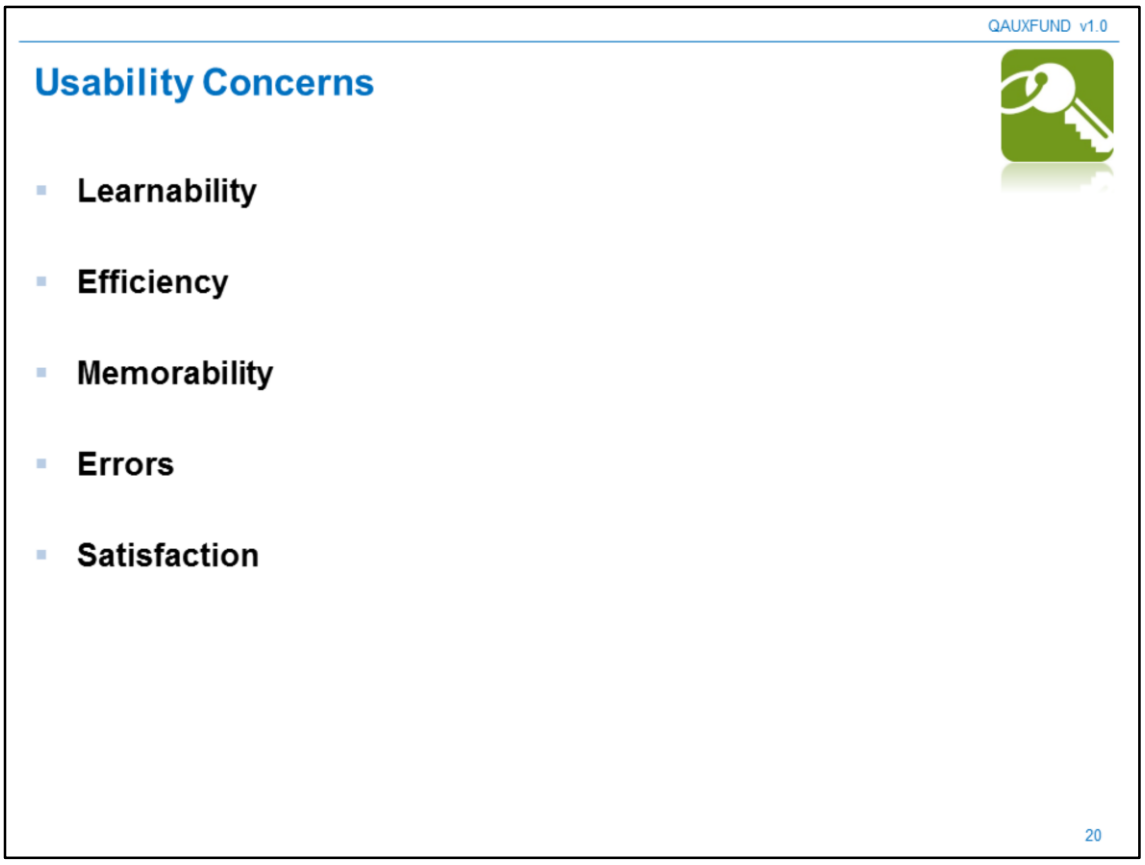
- **Examine the choice we have made in our analysis so far**
  - Take a series of Scenarios and Tasks
  - Consider our User Personas
  - Choose a suitable navigation structure and explain why


QAUXFUND v1.0
Conceptual Models
transforming performance
through learning



The primary notion of usability is that an object designed with a generalized users' psychology and physiology in mind is, for example:

- More efficient to use – takes less time to accomplish a particular task
- Easier to learn – operation can be learned by observing the object
- More satisfying to use



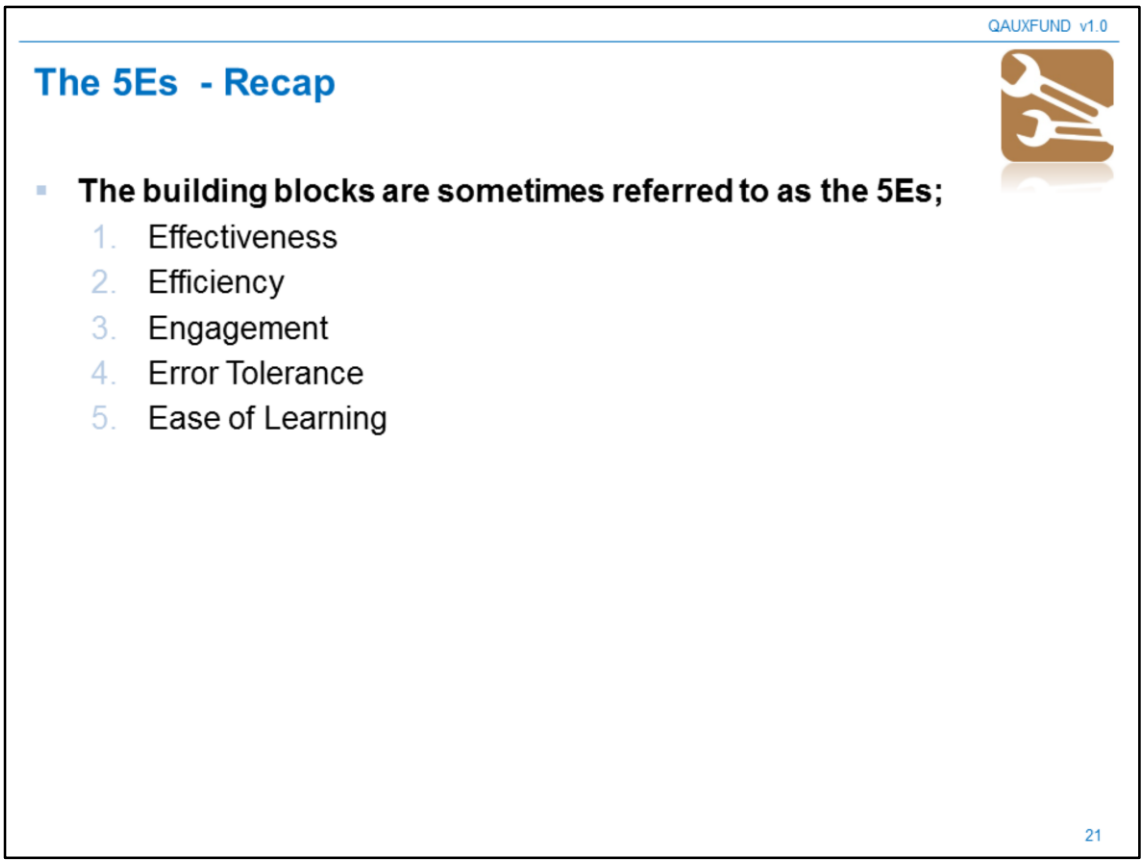
Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?

Efficiency: Once users have learned the design, how quickly can they perform tasks?

Memorability: When users return to the design after a period of not using it, how easily can they re-establish proficiency?

Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?

Satisfaction: How pleasant is it to use the design?



The building blocks of Usability are called the 5Es;

1. Effectiveness

   How completely and accurately the task is completed or the goals are reached

2. Efficiency

   How quickly the task can be completed

3. Engagement

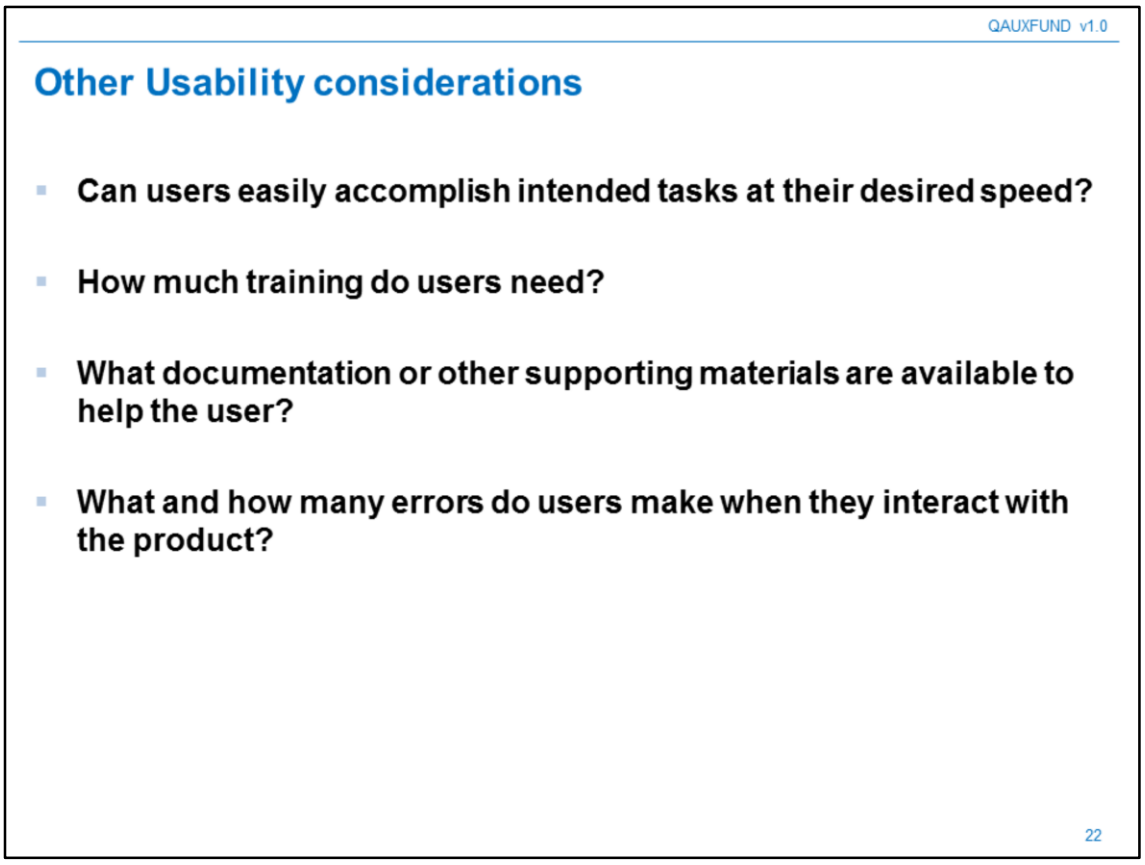
   How effectively the interface draws the user into the interaction and how pleasant and satisfying it is to use

4. Error Tolerance

   How well the product prevents errors and helps the user recover from mistakes

5. Ease of Learning

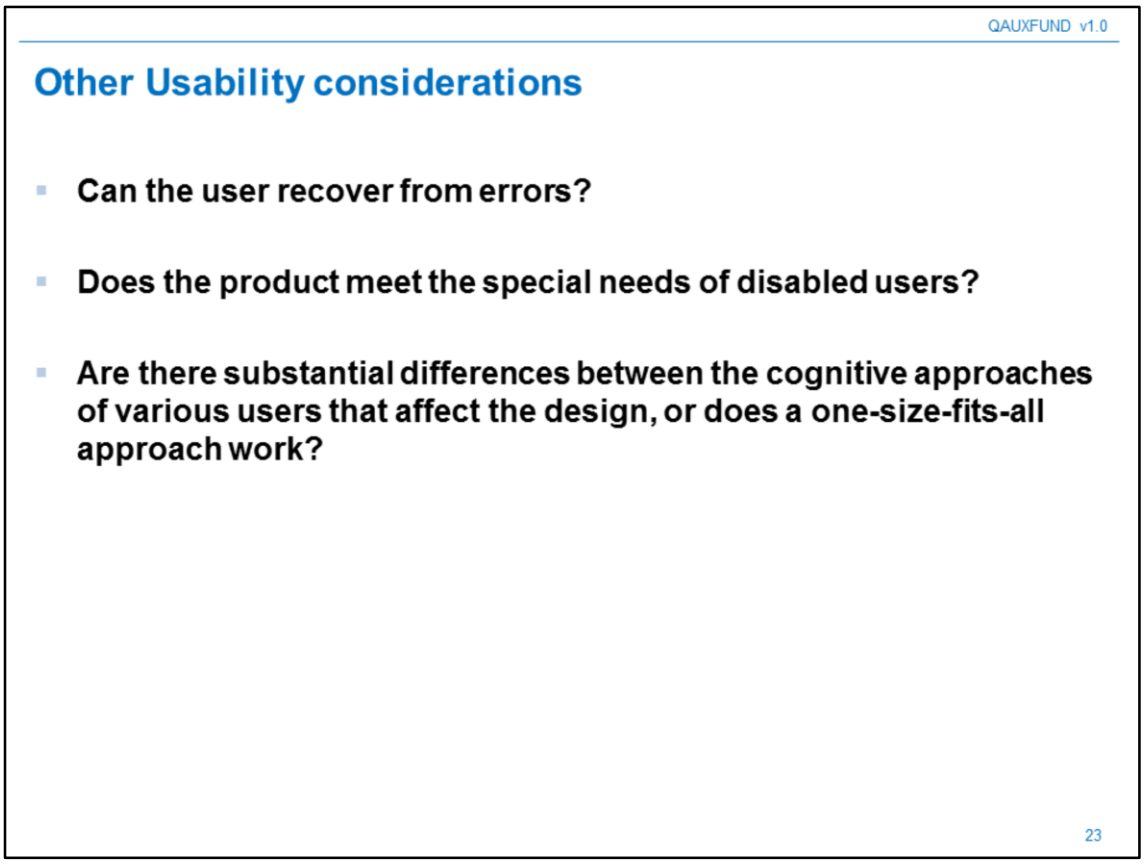
   How well the product supports both initial and continued learning



Other considerations

- Can users easily accomplish intended tasks at their desired speed? **Efficiency**
- How much training do users need?
- What documentation or other supporting materials are available to help the user? **Can users find solutions in these materials?**
- What and how many errors do users make when they interact with the product?
- Can the user recover from errors? What do users have to do to recover from errors? **Does the product help users recover from errors? For example, does software present comprehensible, informative, non-threatening error messages?**
- Does the product meet the special needs of disabled users? **(Is it accessible?)**
- Are there substantial differences between the cognitive approaches of various users that affect the design, or does a one-size-fits-all approach work?

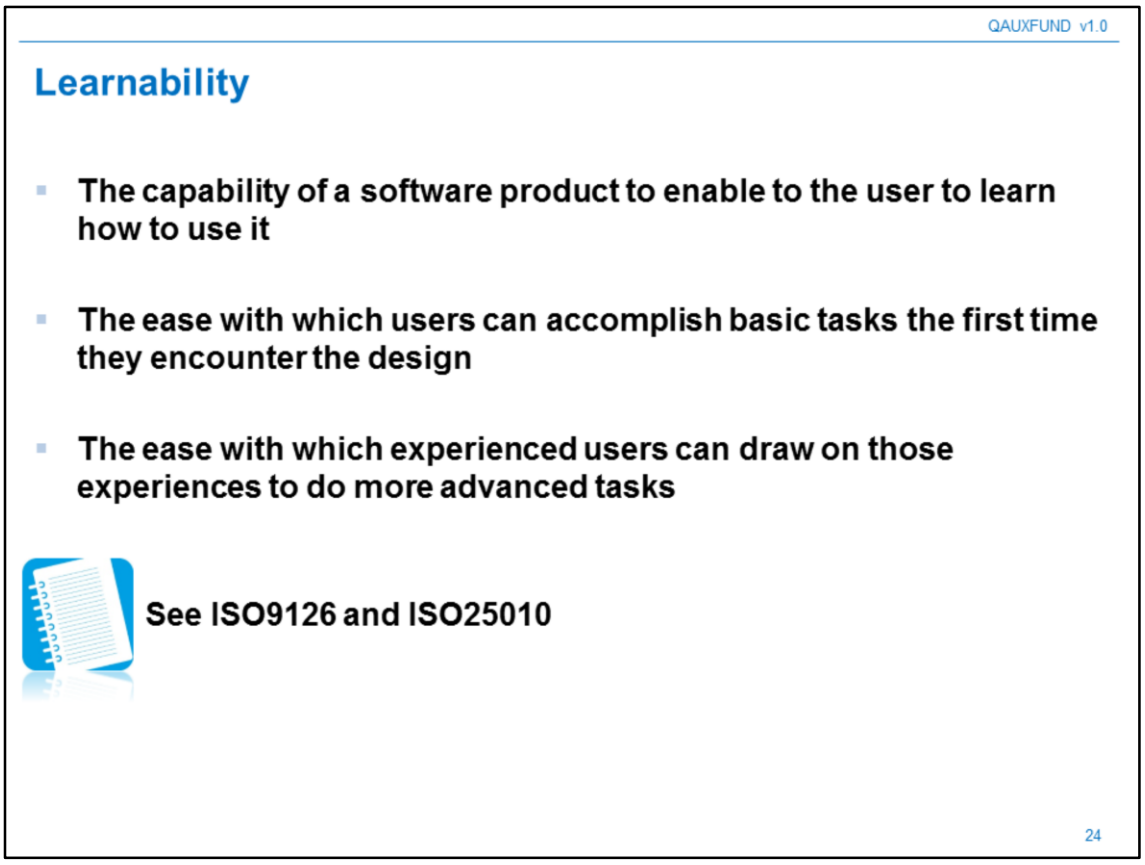
Ways to answer these and other questions include user-focused requirements analysis, building user profiles, and usability testing.



Other considerations

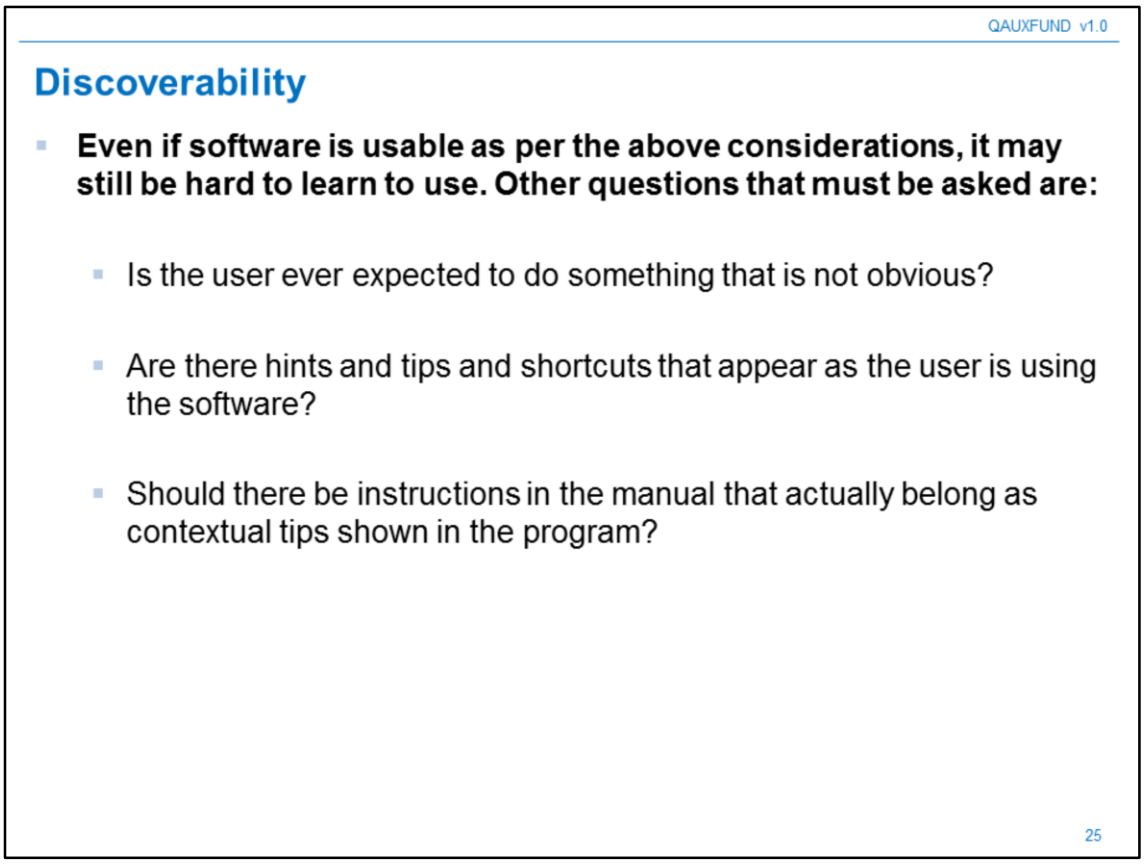
- Can users easily accomplish intended tasks at their desired speed?
- How much training do users need?
- What documentation or other supporting materials are available to help the user? **Can users find solutions in these materials?**
- What and how many errors do users make when they interact with the product?
- Can the user recover from errors? What do users have to do to recover from errors? **Does the product help users recover from errors? For example, does software present comprehensible, informative, non-threatening error messages?**
- Does the product meet the special needs of disabled users? **(Is it accessible?)**
- Are there substantial differences between the cognitive approaches of various users that affect the design, or does a one-size-fits-all approach work?

Ways to answer these and other questions include user-focused requirements analysis, building user profiles, and usability testing.



Learnability is the capability of a software product to enable the user to learn how to use it.

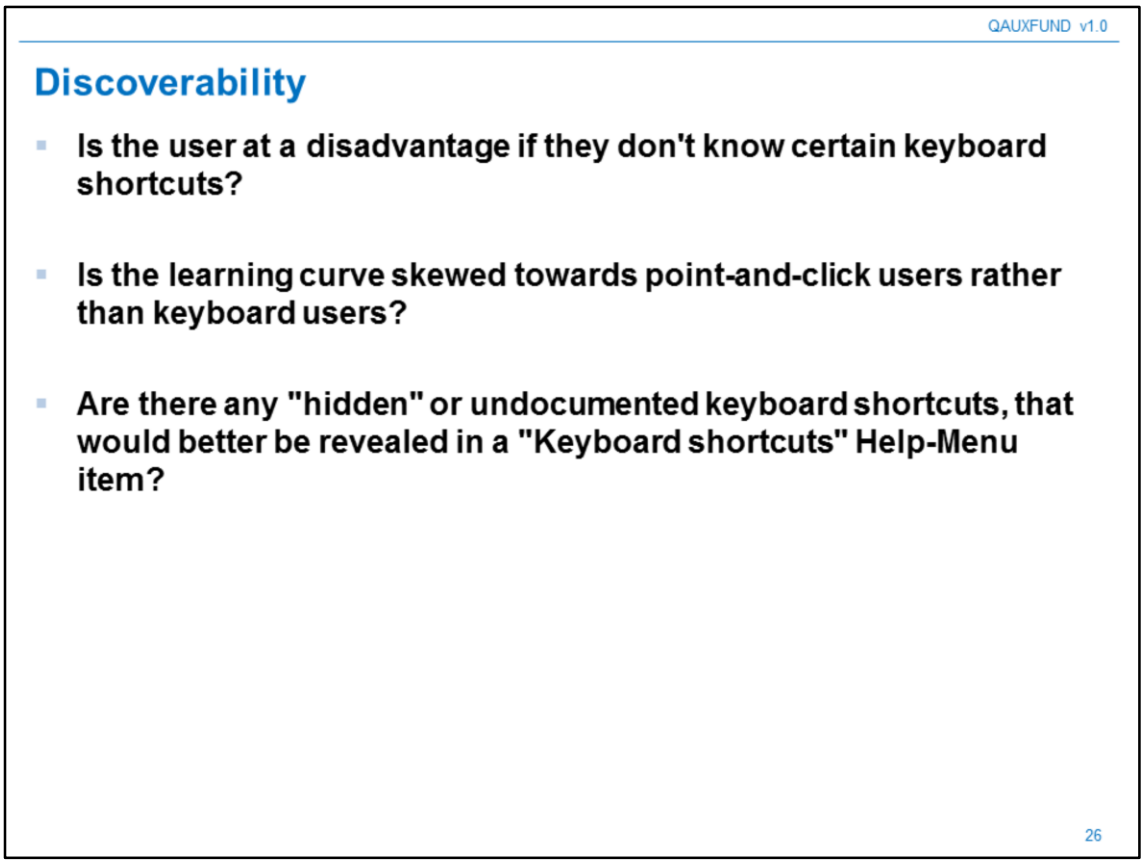
See ISO9126 and ISO25010



Discoverability

Even if software is usable as per the above considerations, it may still be hard to learn to use. Other questions that must be asked are:

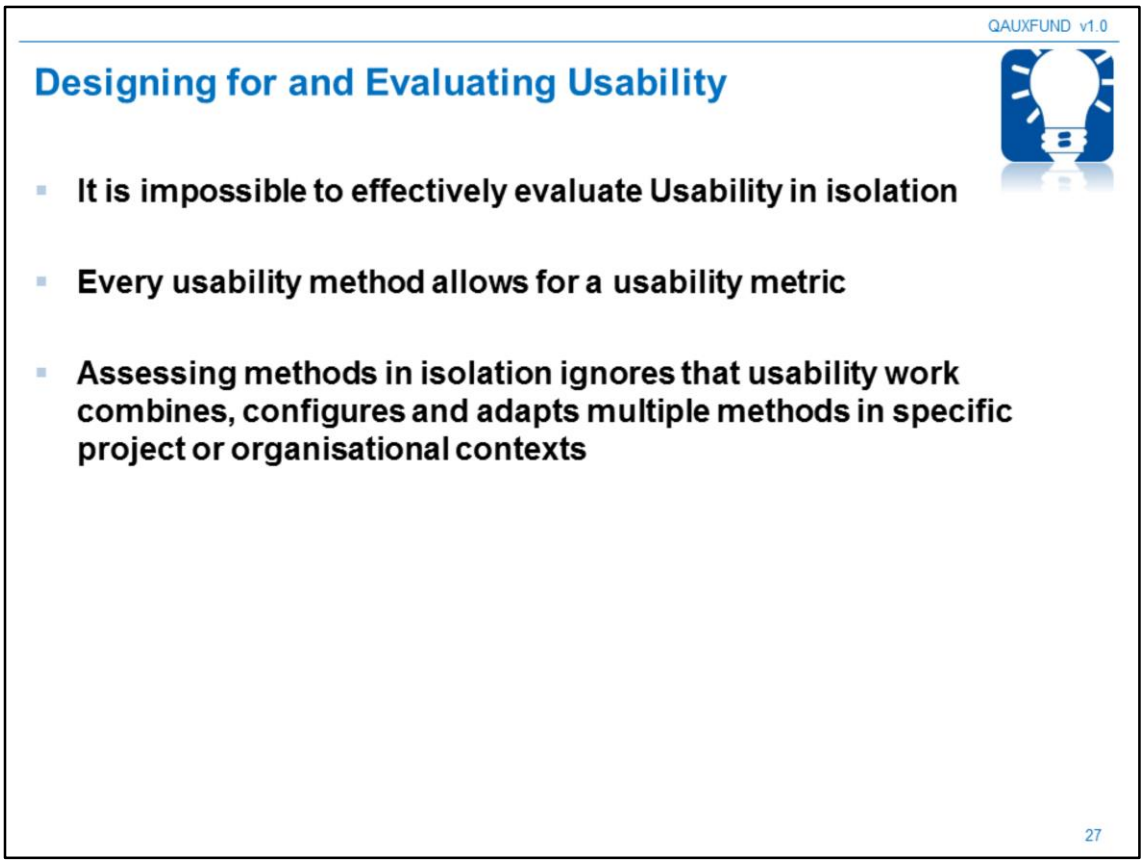
- Is the user ever expected to do something that is not obvious? **(e.g., Are important features only accessible by right-clicking on a menu header, on a text box, or on an unusual GUI element?)**
- Are there hints and tips and shortcuts that appear as the user is using the software?
- Should there be instructions in the manual that actually belong as contextual tips shown in the program?
- Is the user at a disadvantage if they don't know certain keyboard shortcuts? **A user has the right to know all major and minor keyboard shortcuts and features of an application.**
- Is the learning curve (of hints and tips) skewed towards point-and-click users rather than keyboard users?
- Are there any "hidden" or undocumented keyboard shortcuts, that would better be revealed in a "Keyboard shortcuts" Help-Menu item? **A strategy to prevent this "undocumented feature disconnect" is to automatically generate a list of keyboard shortcuts from their definitions in the code.**



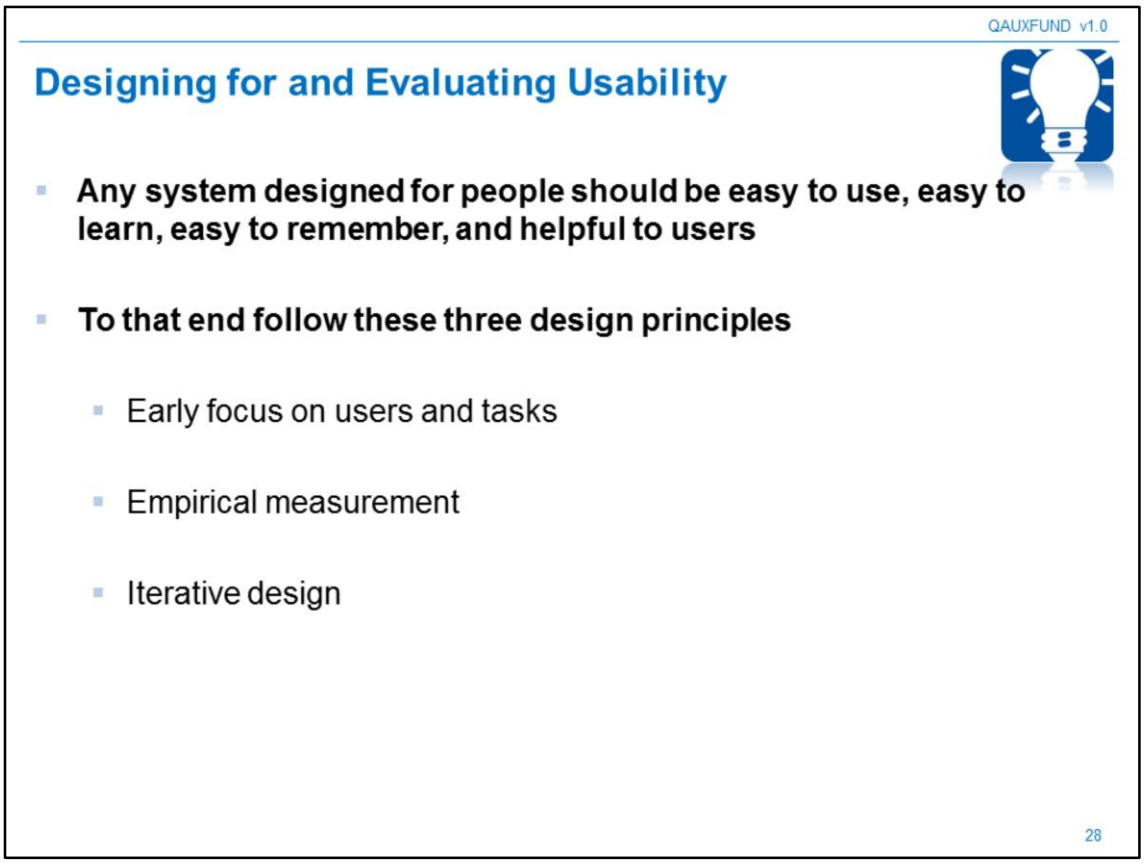
Discoverability

Even if software is usable as per the above considerations, it may still be hard to learn to use. Other questions that must be asked are:

- Is the user ever expected to do something that is not obvious? **(e.g., Are important features only accessible by right-clicking on a menu header, on a text box, or on an unusual GUI element?)**
- Are there hints and tips and shortcuts that appear as the user is using the software?
- Should there be instructions in the manual that actually belong as contextual tips shown in the program?
- Is the user at a disadvantage if they don't know certain keyboard shortcuts? **A user has the right to know all major and minor keyboard shortcuts and features of an application.**
- Is the learning curve (of hints and tips) skewed towards point-and-click users rather than keyboard users?
- Are there any "hidden" or undocumented keyboard shortcuts, that would better be revealed in a "Keyboard shortcuts" Help-Menu item? **A strategy to prevent this "undocumented feature disconnect" is to automatically generate a list of keyboard shortcuts from their definitions in the code.**



A focus on actual work allows realism about design and evaluation methods. Methods are only one aspect of usability work. They are not a separate component of usability work that has deterministic effects, i.e. effects that are guaranteed to occur and be identical across all project and organisational contexts. Instead, broad evaluator effects are to be expected, due to the varying extent and quality of design and evaluation resources in different development settings. This means that we cannot and should not assess usability evaluation methods in artificial isolated research settings. Instead, research should start with the concrete realities of usability work, and within that, research should explore the true nature of evaluation methods and their impact.

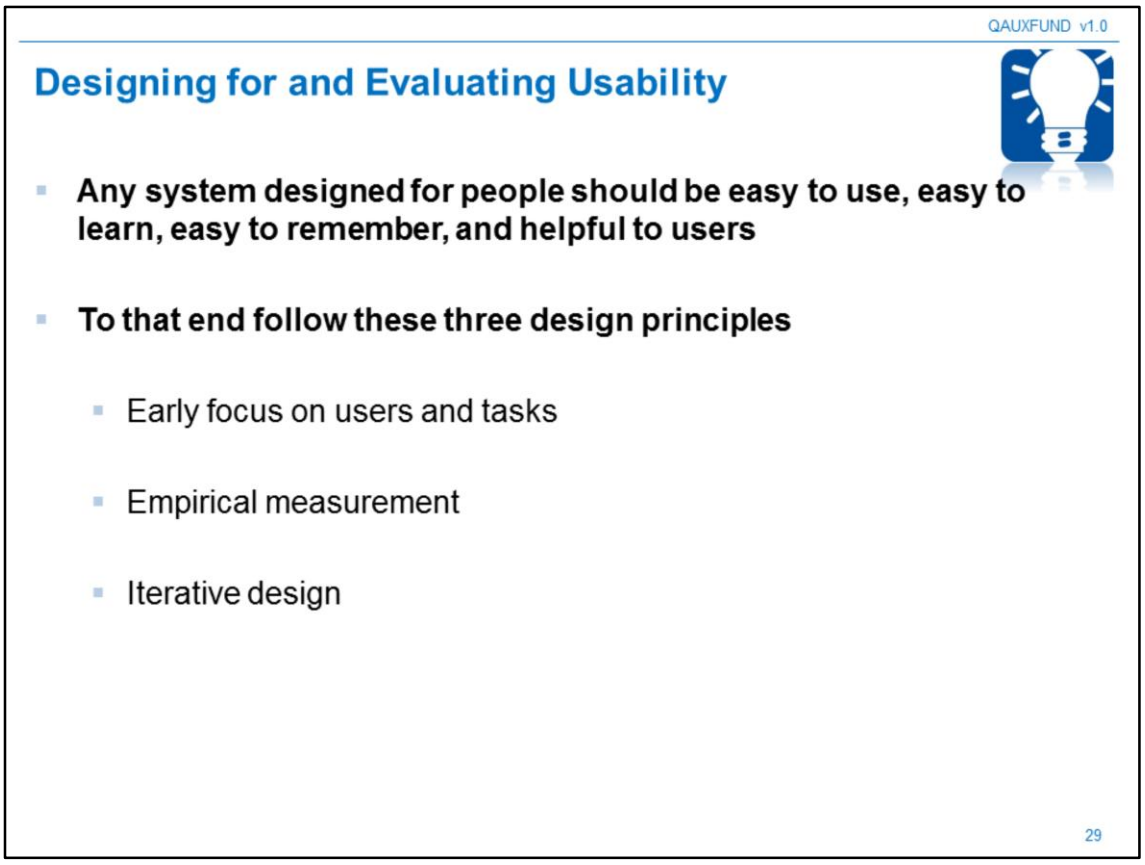


Early focus on users and tasks

The design team should be user driven and in direct contact with potential users. Several evaluation methods, including personas, cognitive modelling, inspection, inquiry, prototyping, and testing methods may contribute to understanding potential users.

Usability considerations such as who the users are and their experience with similar systems must be examined. As part of understanding users, this knowledge must "...be played against the tasks that the users will be expected to perform." This includes the analysis of what tasks the users will perform, which are most important, and what decisions the users will make while using your system. Designers must understand how cognitive and emotional characteristics of users will relate to a proposed system.

One way to stress the importance of these issues in the designers' minds is to use personas, which are made-up representative users. See below for further discussion of personas. Another more expensive but more insightful method is to have a panel of potential users work closely with the design team from the early stages

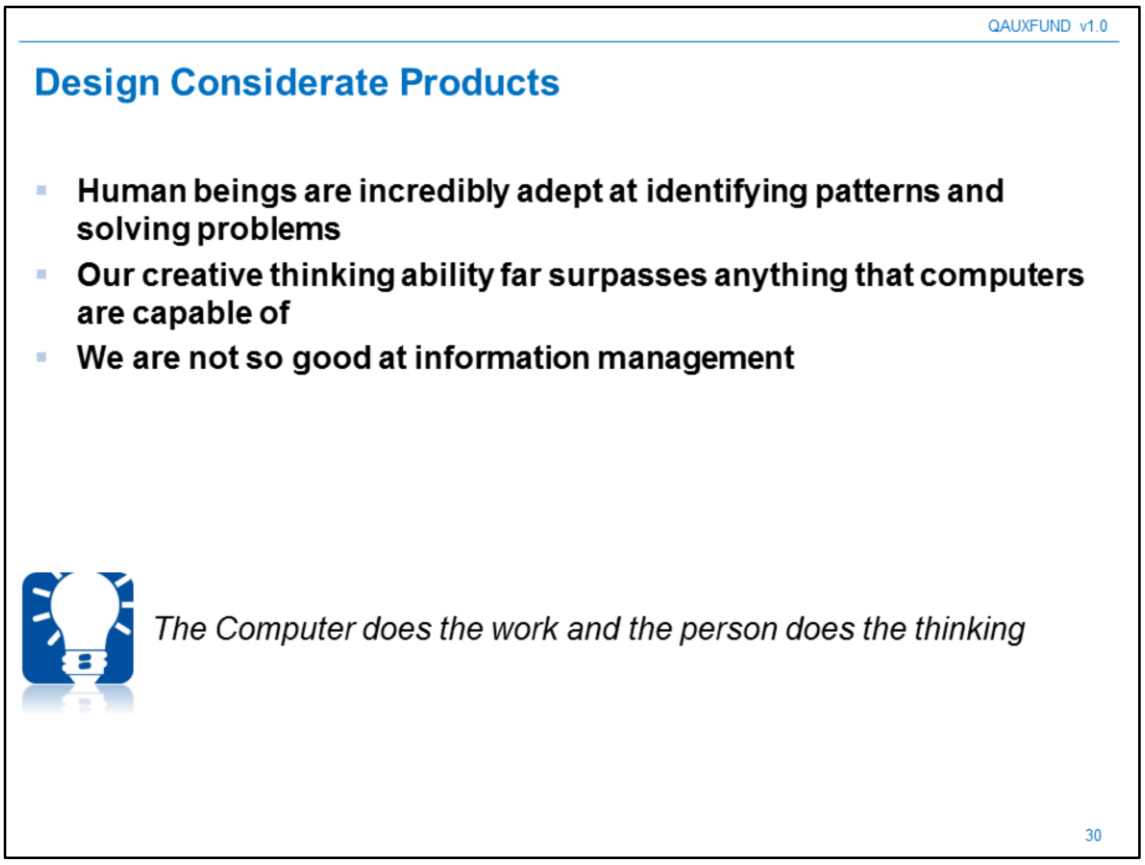


Empirical measurement

Test the system early on, and test the system on real users using behavioural measurements. This includes testing the system for both learnability and usability. (See Evaluation Methods). It is important in this stage to use quantitative usability specifications such as time and errors to complete tasks and number of users to test, as well as examine performance and attitudes of the users testing the system. Finally, “reviewing or demonstrating” a system before the user tests it can result in misleading results. The emphasis of empirical measurement is on measurement, both informal and formal, which can be carried out through a variety of evaluation methods

Iterative design

Iterative design is a design methodology based on a cyclic process of prototyping, testing, analysing, and refining a product or process. Based on the results of testing the most recent iteration of a design, changes and refinements are made. This process is intended to ultimately improve the quality and functionality of a design. In iterative design, interaction with the designed system is used as a form of research for informing and evolving a project, as successive versions, or iterations of a design are implemented. The key requirements for Iterative Design are: identification of required changes, an ability to make changes, and a willingness to make changes. When a problem is encountered, there is no set method to determine the correct solution. Rather, there are empirical methods that can be used during system development or after the system is delivered, usually a more inopportune time. Ultimately, iterative design works towards meeting goals such as making the system user friendly, easy to use, easy to operate, simple, etc.



The Computer does the work and the person does the thinking

Human beings are incredibly adept at identifying patterns and solving problems.

Our creative thinking ability far surpasses anything that computers are capable of.

We are not so good at information management.

Software should be considerate. It should be concerned with the needs of others (specifically the users) beyond the considerations of performing basic functions.

- Do not obscure the process
- Do not force users to hunt for common functions
- Do not lay blame for mistakes and failings

- Do provide lots of relevant information.
- Do be helpful.
- Do be respectful towards users.

Software should behave like a considerate human being

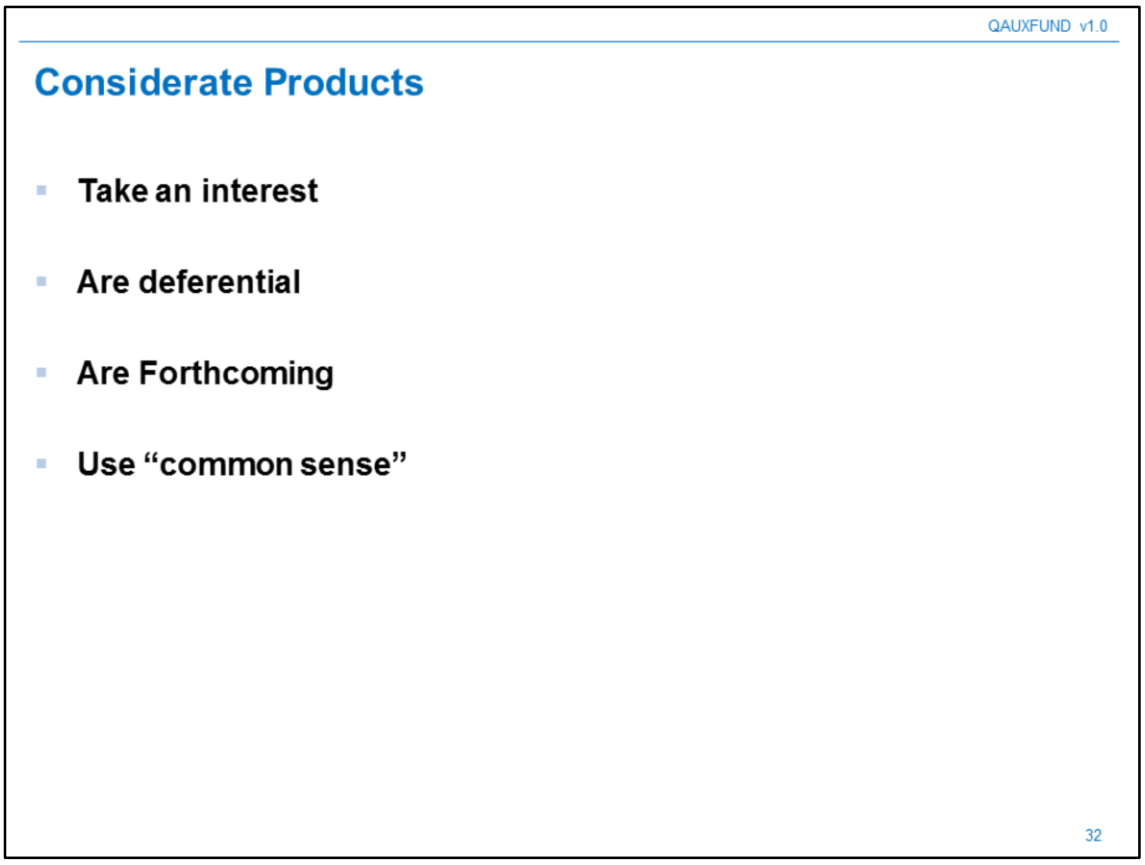
Design Considerate Products

- **Software should be considerate, concerning itself with the needs of others beyond the considerations of performing basic functions**

- **DO**
  - Provide lots of relevant information
  - Be helpful
  - Be respectful towards users

- **DO NOT**
  - Obscure the process
  - Force users to hunt for common functions
  - Lay blame for mistakes and failings

Software should behave like a considerate human being



The most important characteristics of considerate products are:

- They take an interest

  Software should work hard to remember our habits.

  Software is much better at remembering than humans are. If we tell the software something it should remember it, and work on the assumption that the information remains the same next time it needs to ask, unless told otherwise.

- They are deferential

  Do not pass judgement on human actions.

  At most, suggest that there may be a mistake, and explain the consequences should we choose to continue. Do not bar the user from continuing.

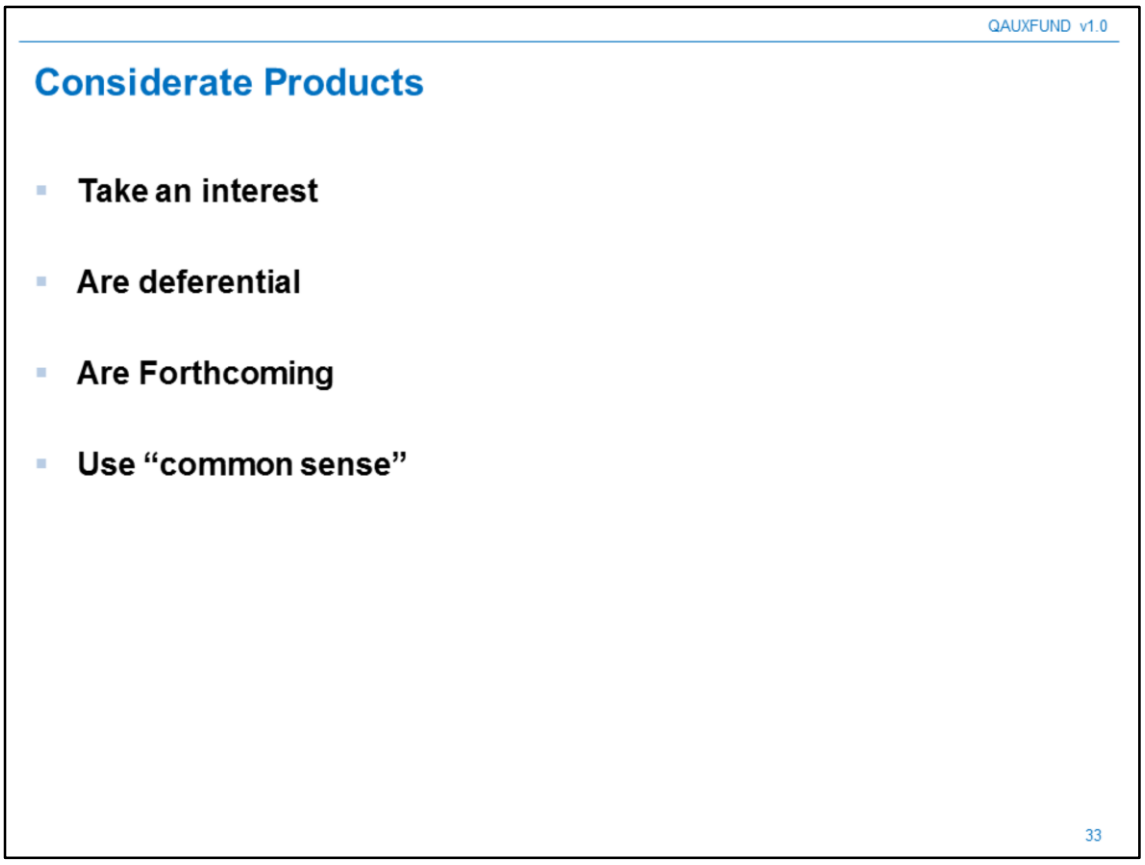
  Remember that the computer submits to the user, not the other way around.

  In fact, avoid using “submit” on buttons. You are telling the user to submit.

- They are forthcoming

  Most software does not attempt to provide related information. Instead it narrowly answers the precise questions we ask it, failing to provide data that is clearly relevant to the user’s goals.

  It requires a delicate touch. “Clippy” is this principle taken to the extreme.



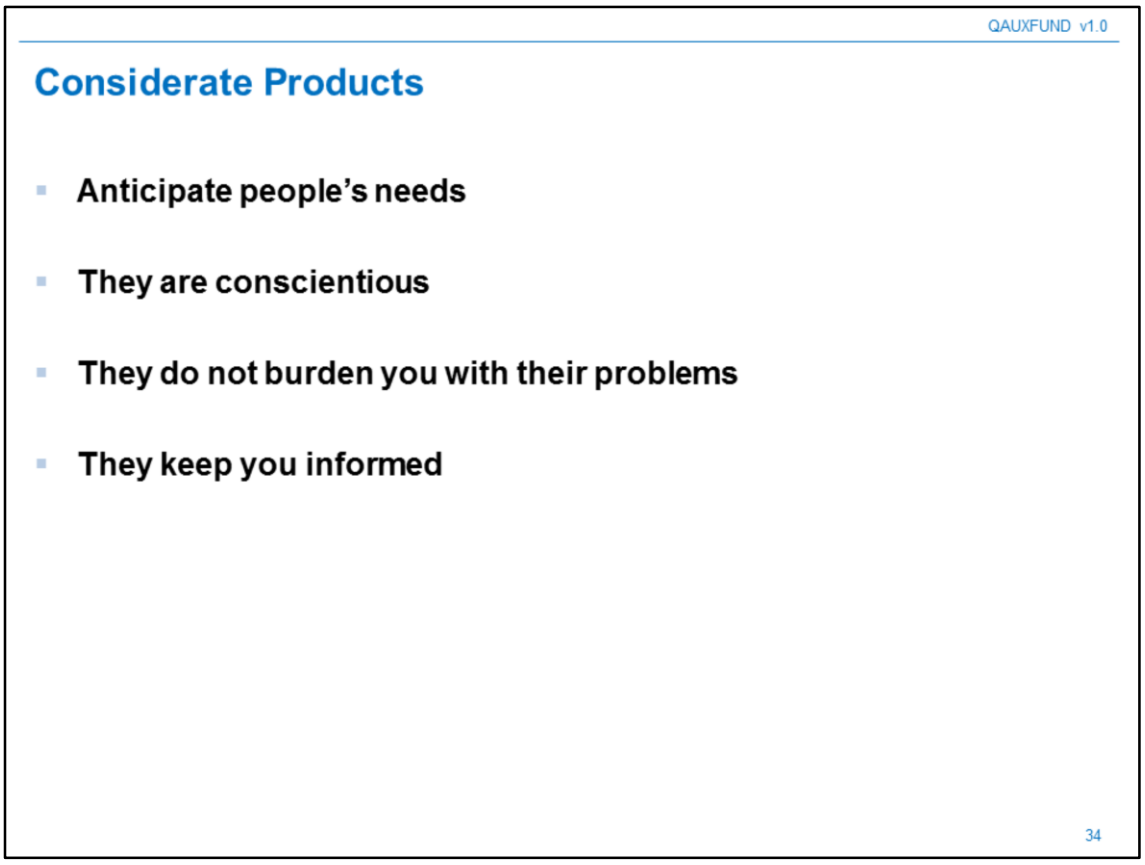
- They are forthcoming (cont.)

  It should be like a good waiter; they refill your glass when it is empty, but do not hang around when you do not need them.

- They use “common sense”

  Do not put the controls for rarely or never used functions next to constantly used ones.

  Do not automatically act on data that would look ridiculous to a human being; e.g. do not send a cheque for £0.00, or bill someone thousands of pounds for a phone bill without checking it with a human being first



- They anticipate people's needs

  Anticipate possible requests, and start preparing responses to them before they are made.

  e.g. a web browser should start to preload the visible links, or a flight booking system should provide seating options, confirm luggage requirements, and maybe even hotel booking options.

- They are conscientious

  Do not overwrite data unless specifically instructed to

  If data overlaps, find some way of saving both, and making sure both are distinct, separate, and findable.

  e.g. You save a new document with a filename that already exists. The software should allow you to rename one or both files rather than deleting the old file to replace with the new.

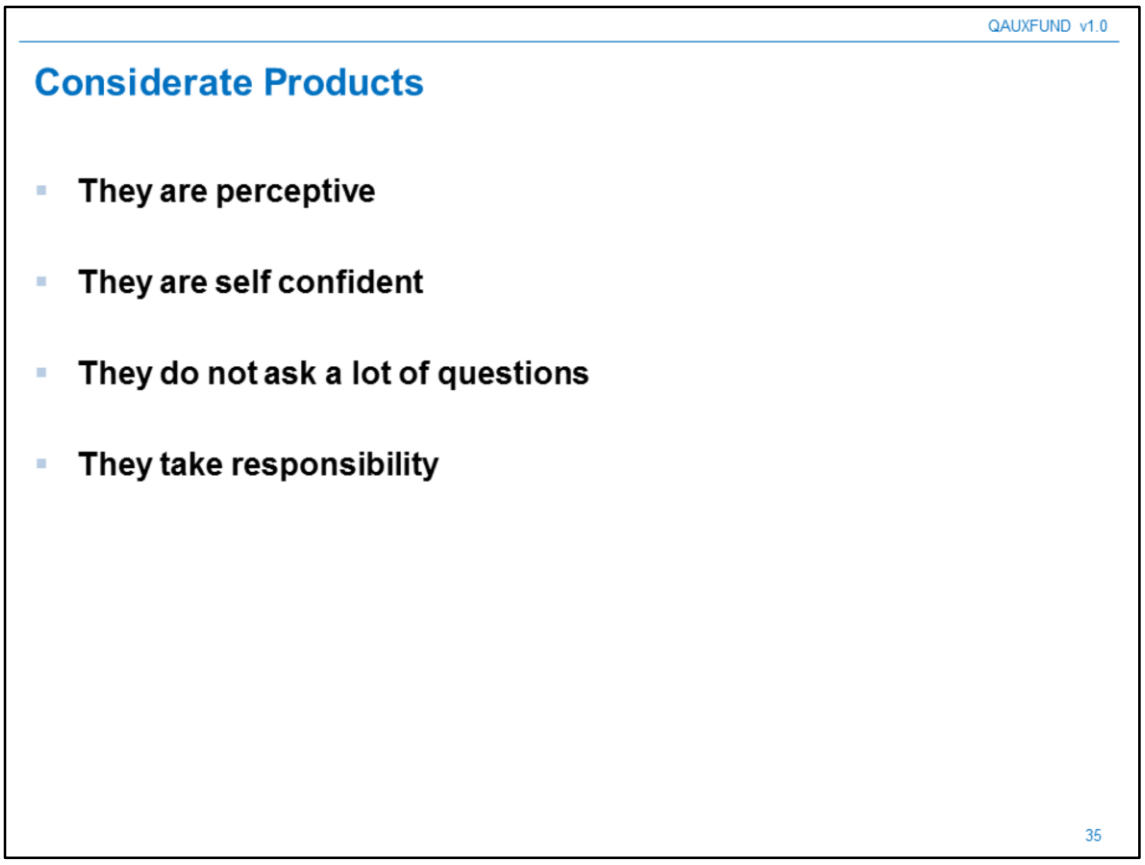
- They do not burden you with their problems

  No unnecessary error notifications

  Find ways of making the necessary ones, unnecessary.

- They keep you informed

  Provide all information needed in plain sight, at the site where it is needed



- They are perceptive

  Products should watch our preferences and remember them without being explicitly asked to do so.

  e.g. window size and location information.

- They are self confident

  Do not second guess the user

  If the computer has any reason to suspect that the user might have made a mistake (e.g. they have deleted a file) it should anticipate them changing their minds by (in this example) undeleting the file.

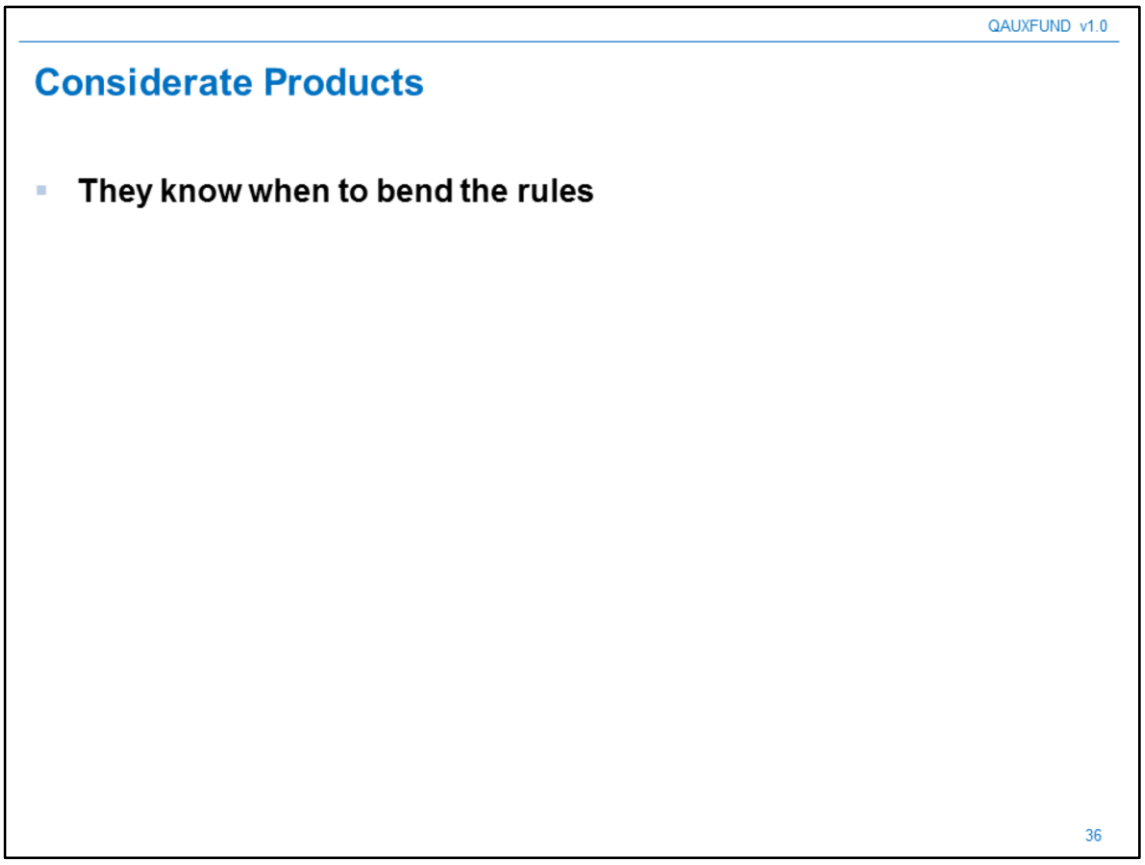
- They do not ask a lot of questions

  No choices should be forced upon the user

  They are given options, which they are free to select, or not.

- They take responsibility

  They can fail or degrade gracefully.

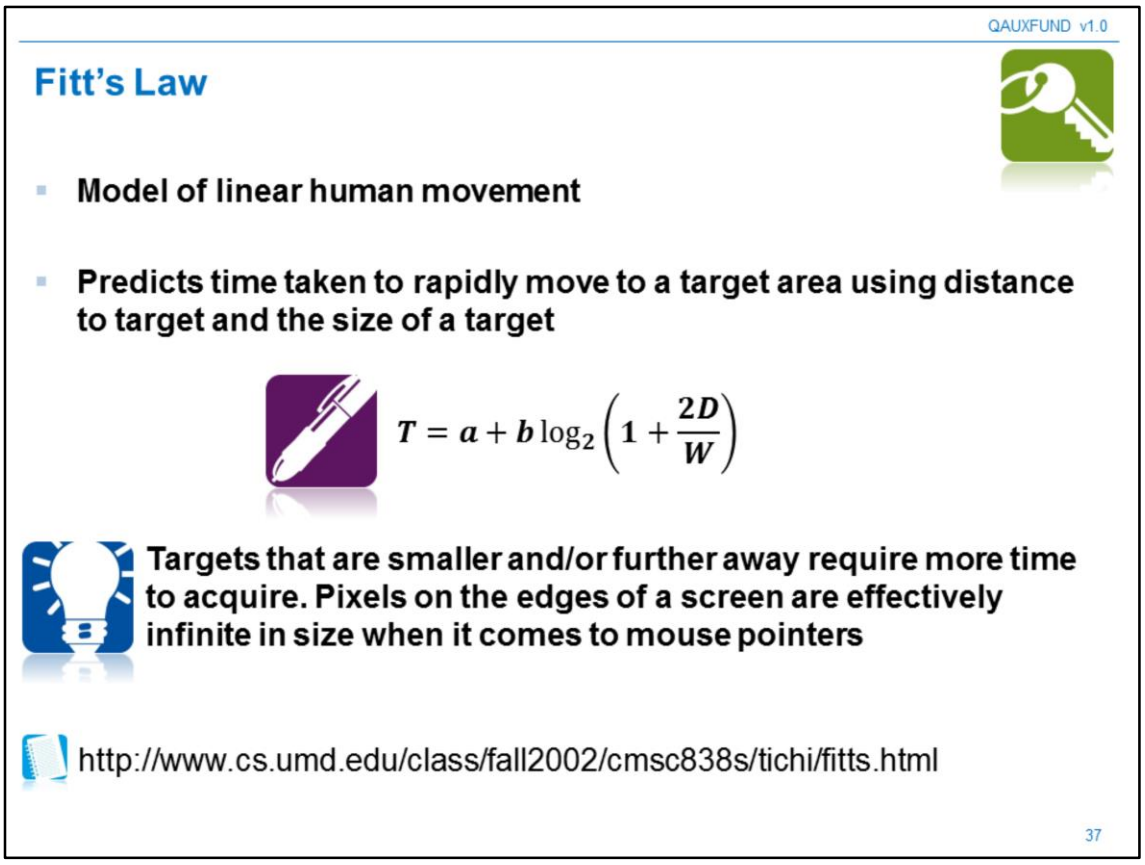


- They know when to bend the rules

  Your system needs a delivery address, but your product is being bought by a company that is in the process of moving location but they are still finalising their decision on which potential office they will be moving to in twelve months time.

  They need to order the equipment you are selling now due to long manufacture times for the volume required, but can not confirm which of the three offices they are looking at will be the delivery address

  Your system needs to be able to bend the rules and allow them to order without a delivery address.



T is the average time taken to complete the movement. (Traditionally, researchers have used the symbol *MT* for this, to mean *movement time*.)

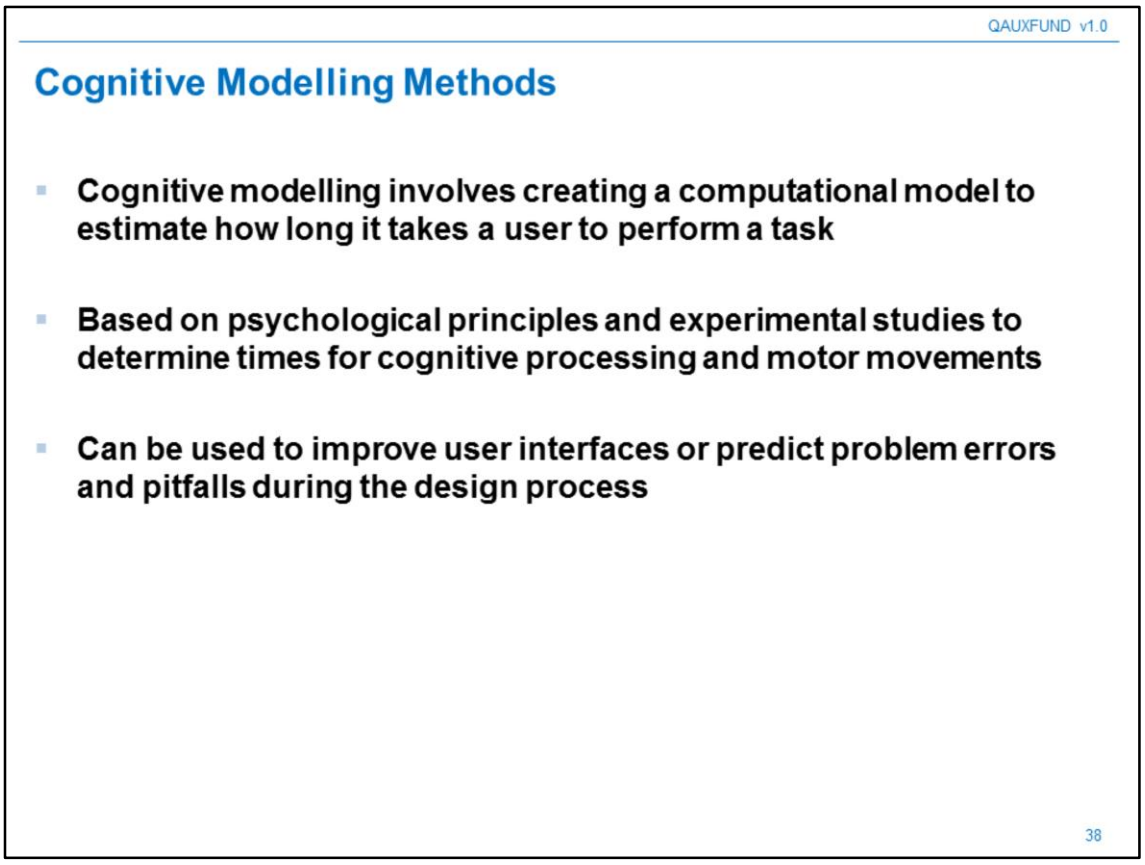
a represents the start/stop time of the device (intercept) and

b stands for the inherent speed of the device (slope). These constants can be determined experimentally by fitting a straight line to measured data.

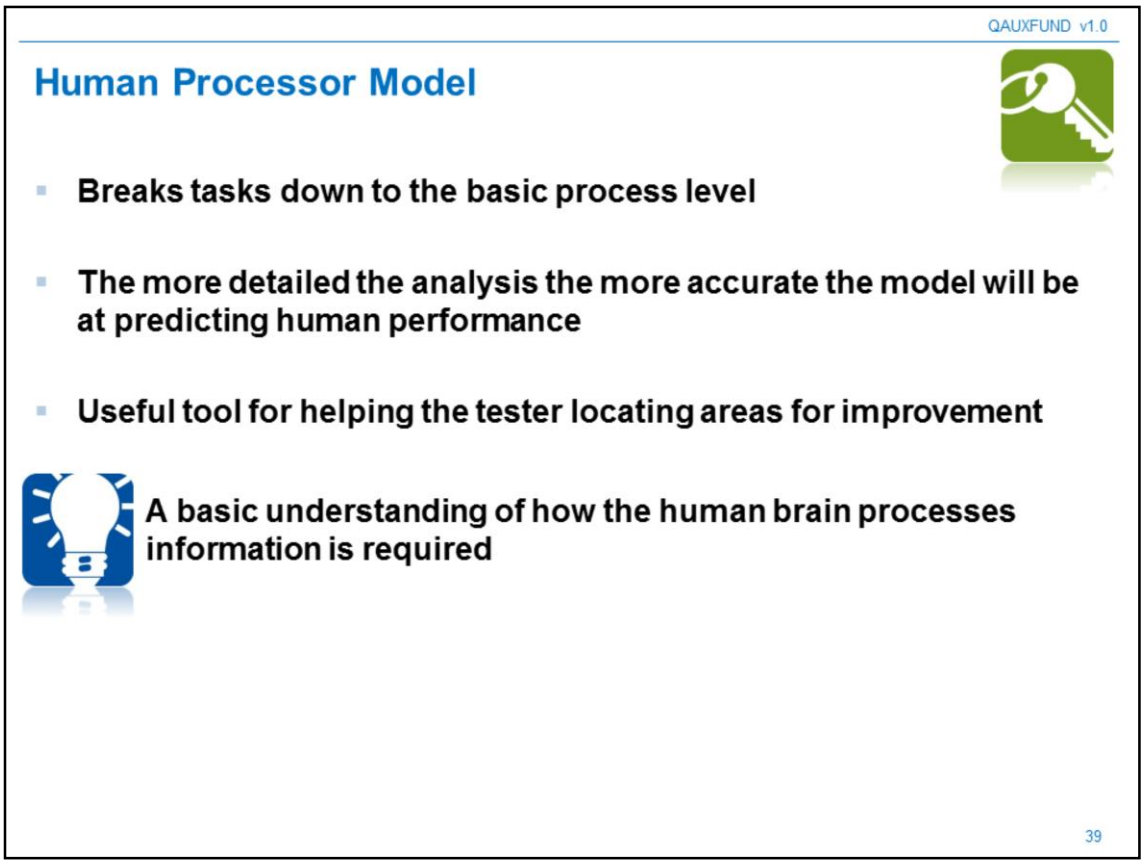
D is the distance from the starting point to the centre of the target. (Traditionally, researchers have used the symbol *A* for this, to mean the *amplitude* of the movement.)

W is the width of the target measured along the axis of motion. *W* can also be thought of as the allowed error tolerance in the final position, since the final point of the motion must fall within $\pm W/2$ of the target's center.

From the equation, we see a *speed–accuracy* trade off associated with pointing, whereby targets that are smaller and/or further away require more time to acquire.

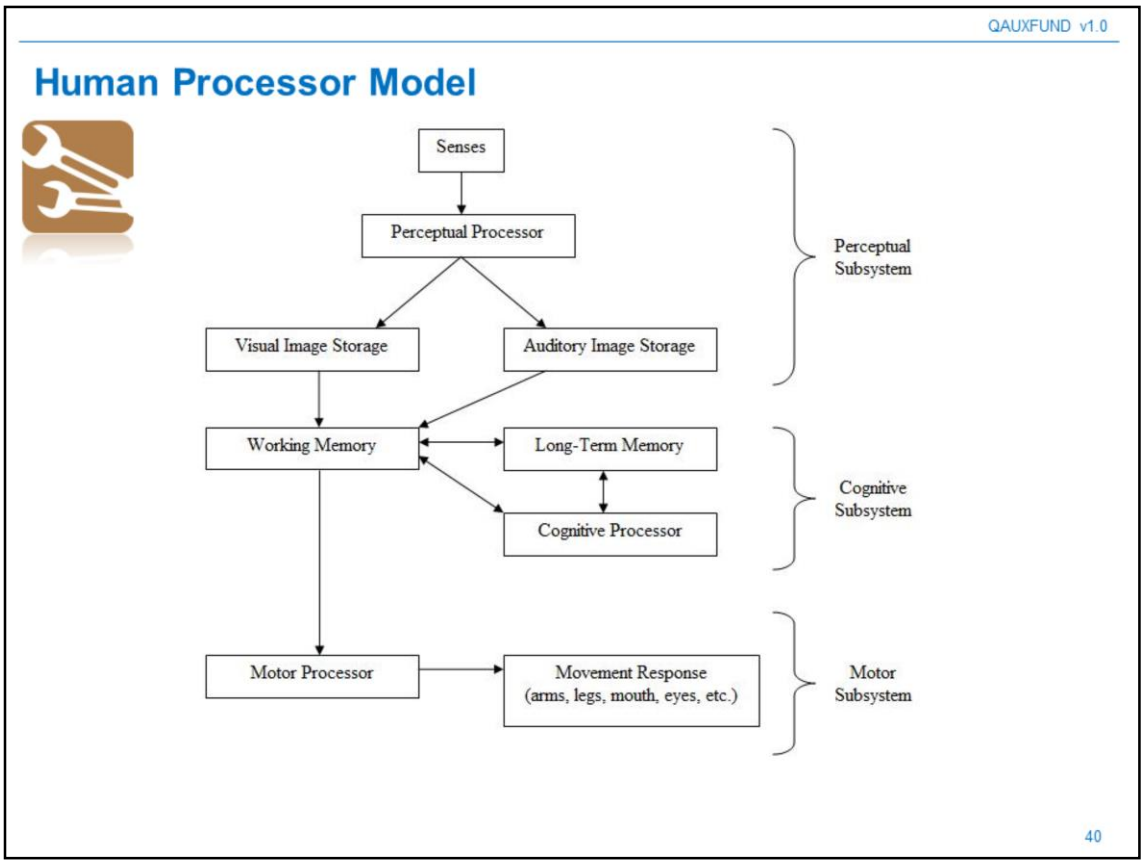


Cognitive modelling involves creating a computational model to estimate how long it takes people to perform a given task. Models are based on psychological principles and experimental studies to determine times for cognitive processing and motor movements. Cognitive models can be used to improve user interfaces or predict problem errors and pitfalls during the design process. A few examples of cognitive models include:



Human Processor Model

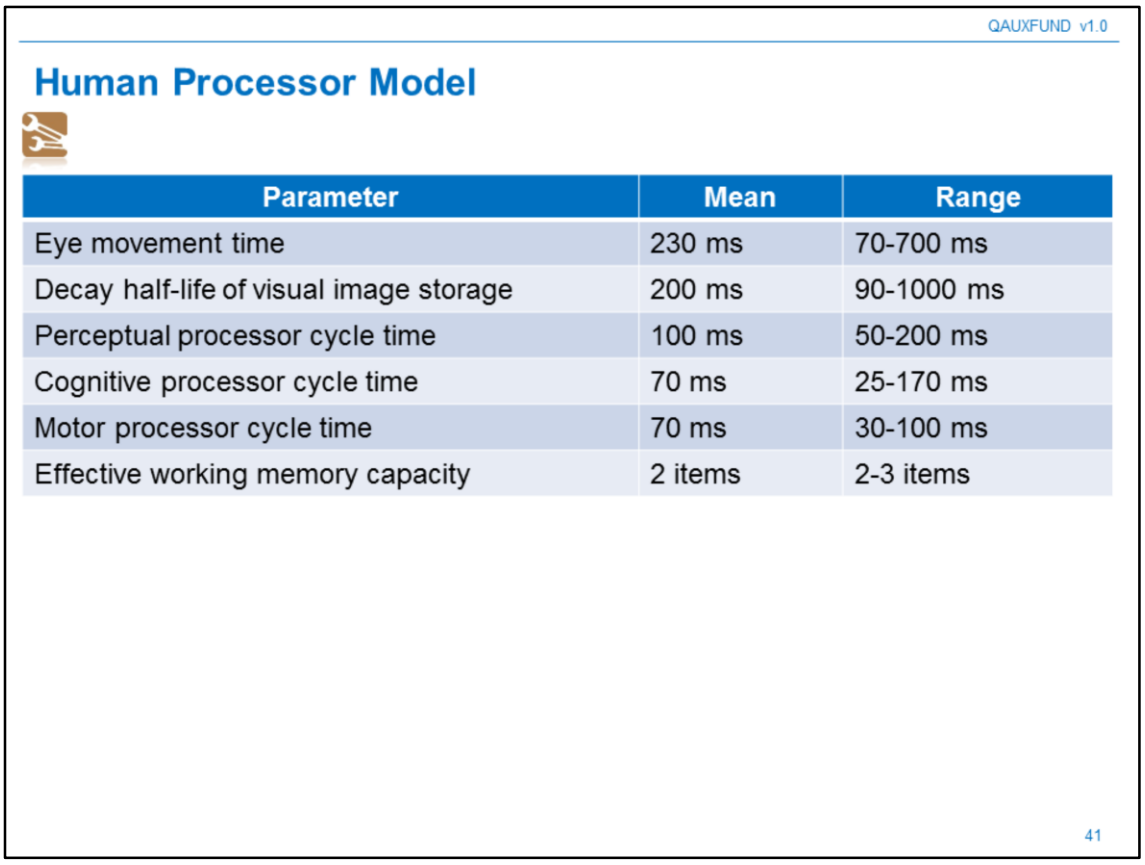
Sometimes it is useful to break a task down and analyse each individual aspect separately. This helps the tester locate specific areas for improvement. To do this, it is necessary to understand how the human brain processes information.



Human Processor Model

Sometimes it is useful to break a task down and analyse each individual aspect separately. This helps the tester locate specific areas for improvement. To do this, it is necessary to understand how the human brain processes information.

The human processor model uses the cognitive, perceptual, and motor processors along with the visual image, working memory, and long term memory storages. A diagram is shown below. Each processor has a cycle time and each memory has a decay time. These values are also included below. By following the connections diagrammed below, along with the associated cycle or decay times, the time it takes a user to perform a certain task can be calculated.



Many studies have been done to estimate the cycle times, decay times, and capacities of each of these processors. Variables that affect these can include subject age, aptitudes, ability, and the surrounding environment. These results are reasonable estimates for a younger adult

Long-term memory is believed to have an infinite capacity and decay time

How to Calculate

Write out main steps based on: a working prototype, simulation, step by step walk-through of all steps

Clearly identify the specific task and method to accomplish that task

For each final step identify sub-levels down to a basic process (in the diagram or chart below)

Convert into pseudo code (writing out methods for each step)

List all assumptions (will be helpful as multiple iterations are completed)

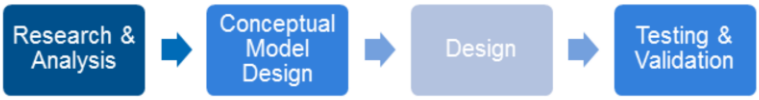
Determine time of each operation (based on the table below)

Determine if operation times should be adjusted (slower for elderly, disability, unfamiliarity, etc.)

Sum up execution times

Iterate as needed and check with prototyping if possible

Conceptual Models



42

QAUXFUND v1.0

Mental Model vs. Conceptual Model

A mental model is the representation that a person has in his mind about the object he is interacting with

A conceptual model is the actual model that is given to the person through the design and interface of the actual product

100 Things Every Designer Needs to Know About People
Susan M. Weinschenk. 2011.

43

QAUXFUND v1.0

Conceptual models – Google Books

Conceptual models – Southwest Airlines

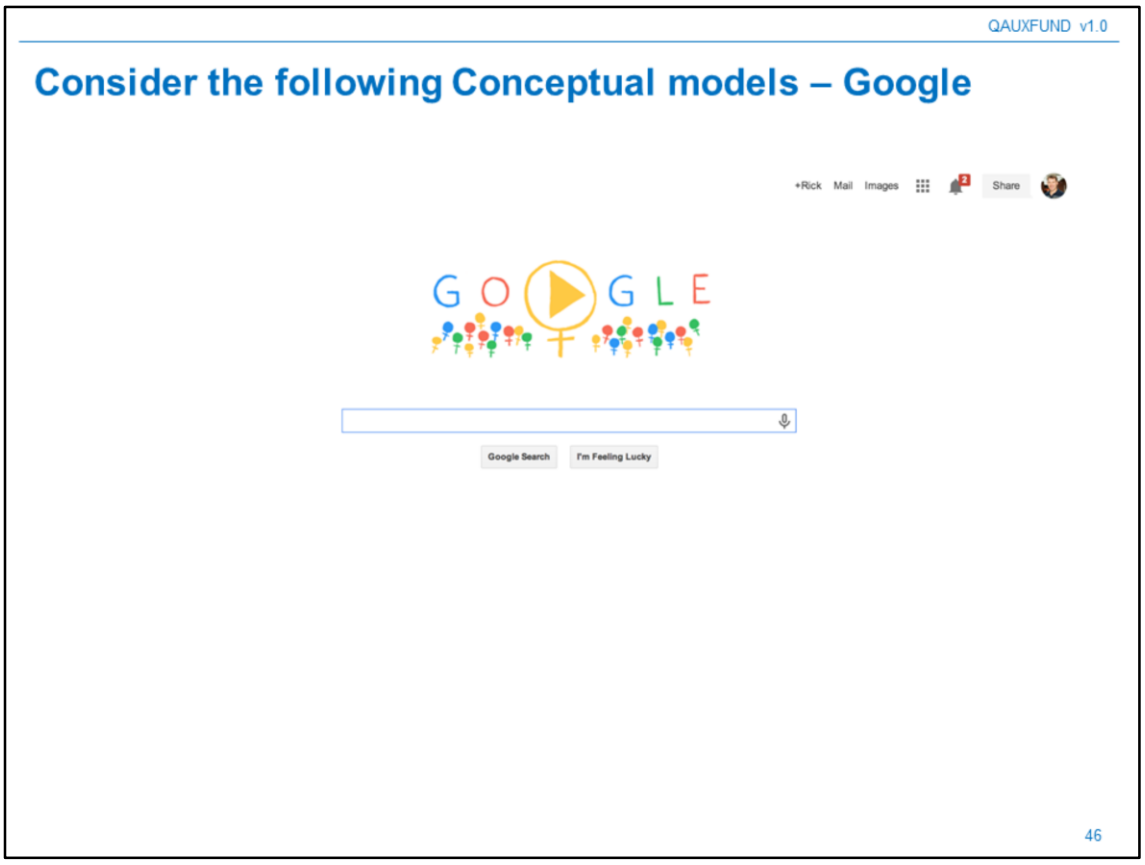


Image 1 – Google 1998

Image 2 – Google 2000

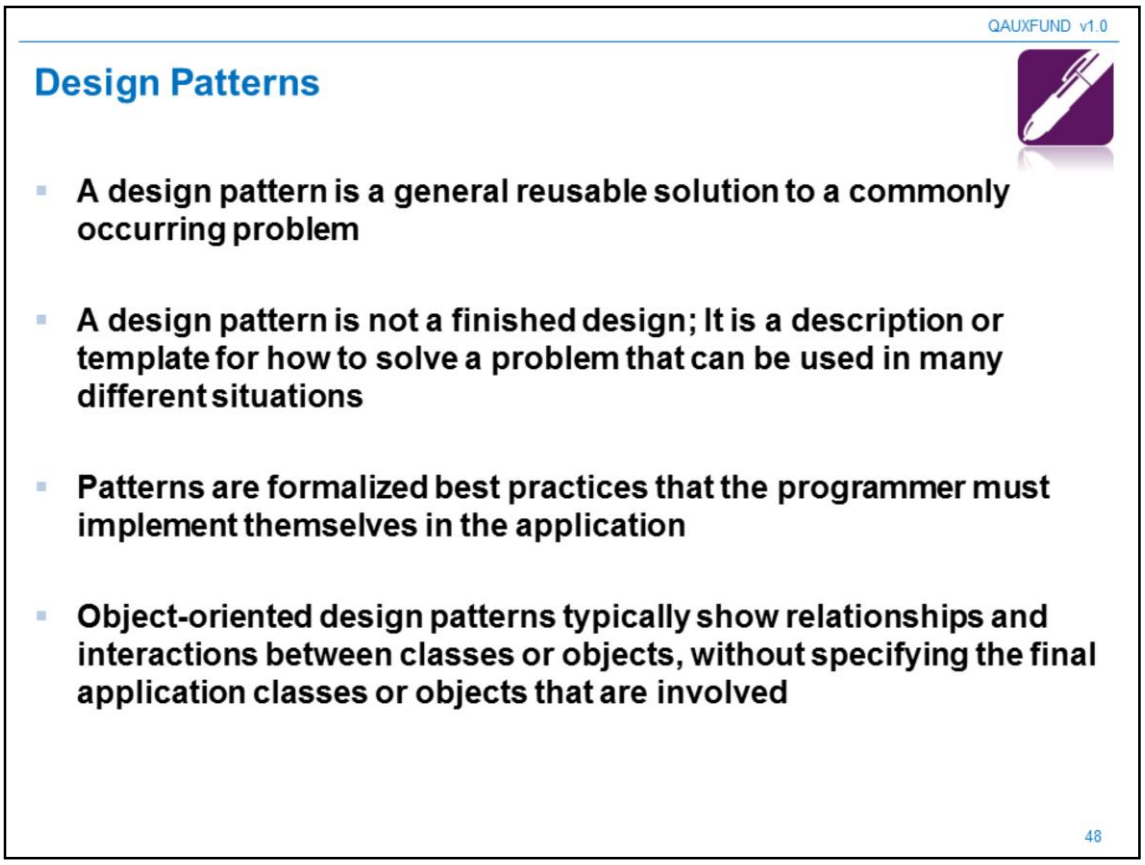
Image 3 – Google 2013

QAUXFUND v1.0

Group Exercise – Cinema Booking

- **In small groups examine three major cinema groups**
  - Explore the conceptual model and journey through to booking a ticket

47



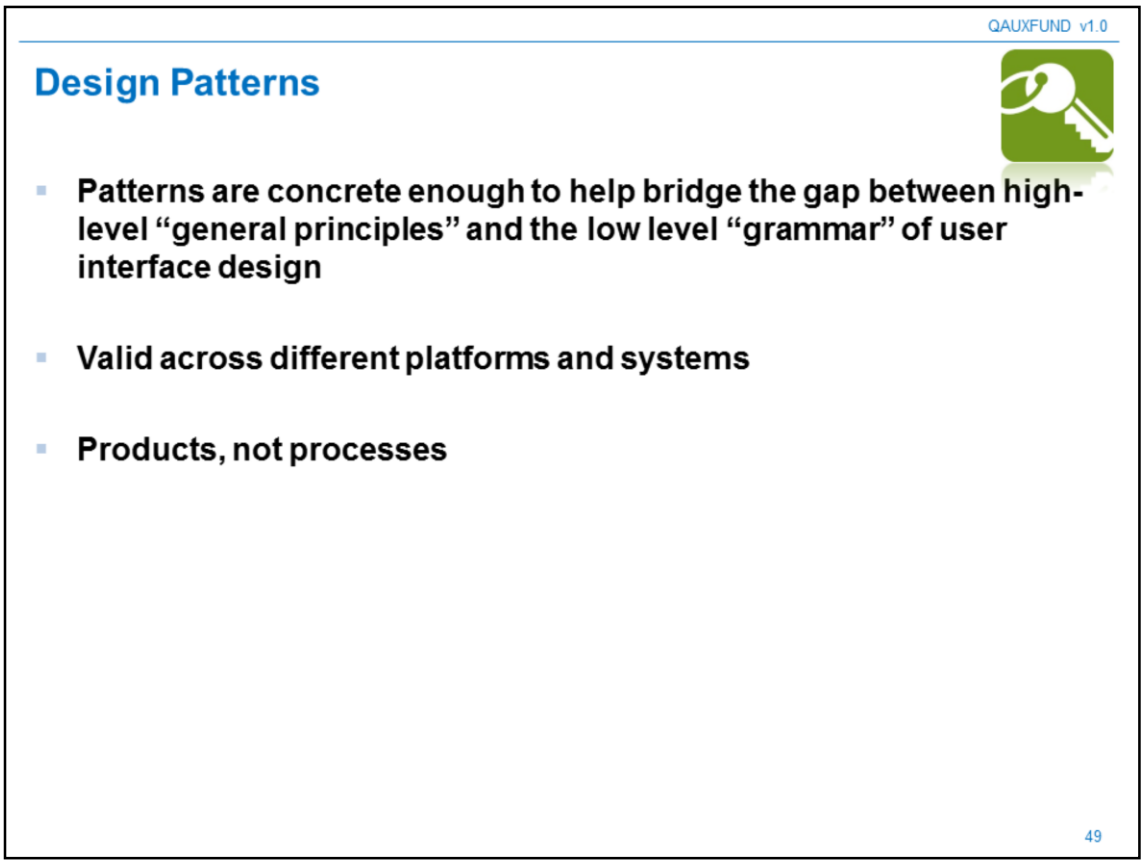
What is a design Pattern?

It is a design that is used again and again, in order to create consistency. It describes a design in terms of standards that are used throughout the application, or across multiple applications.

Broadly speaking, they are small interface pieces that are loosely joined together into familiar groups. They describe the structural and behavioural features in a way that makes them easier to understand and make the tools more useful.

More often than not, the design pattern describes the layout, but usually also includes the rules of interactive elements.

Some patterns (especially website patterns) are industry standards, where others are much more limited in their areas of use. E.g. the MS Office Ribbon is a design pattern that is used across all Microsoft Office applications, but is rarely used by third parties.



Patterns are:

- Concrete

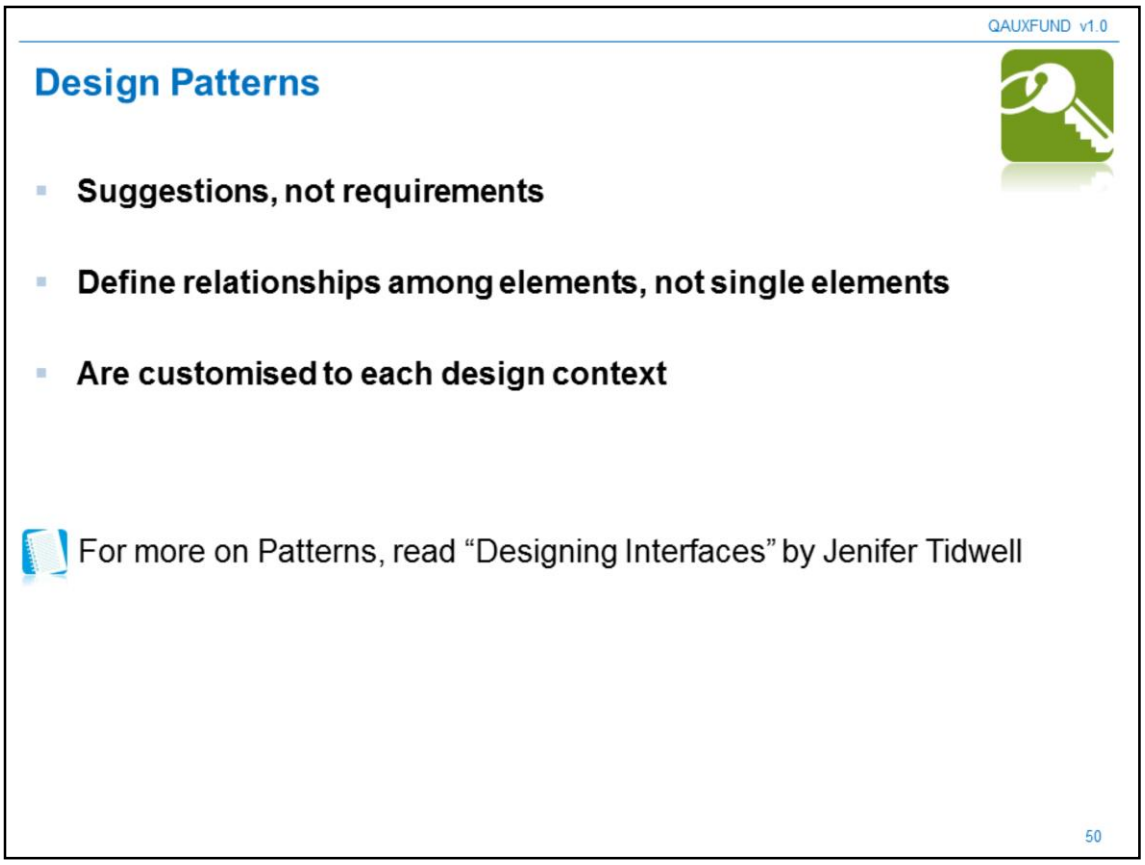
  Patterns are concrete enough to help bridge the gap between high-level “general principles” and the low level “grammar” of user interface design, such as widgets, text, graphic elements, alignment grids etc..

- Valid across different platforms and systems

  Patterns are more concrete than principles or heuristics, but they do define abstractions where the best patterns are not specific to a single platform or idiom. Ideally each pattern remains relevant even when the underlying technologies and media change.

- Products, not processes

  Unlike heuristic or user-centred design techniques which usually focus on the process of finding a solution, patterns are pre-defined solutions that it may be possible to apply to a problem.



- Suggestions, not requirements

  Patterns are only intended to be suggestions that you can follow or reject. Their suitability depends on your design context and the needs of the user.

- Relationships among elements, not single elements

  A text field is not a pattern. A two panel layout where selections are made in the first panel, and information related to the selection are displayed in a text field inside the second panel might however be a pattern.

  Changes in a set of elements over time (as the user interacts with the software) can also constitute a pattern, although not all patterns capture changes are may deal with purely static relationships.

- Customised to each design context

  When a pattern is used in a design, the designer must adjust the pattern to fit the particular users and requirements.

We can not hope to cover the range of patterns that might be used, so here are a few common examples.

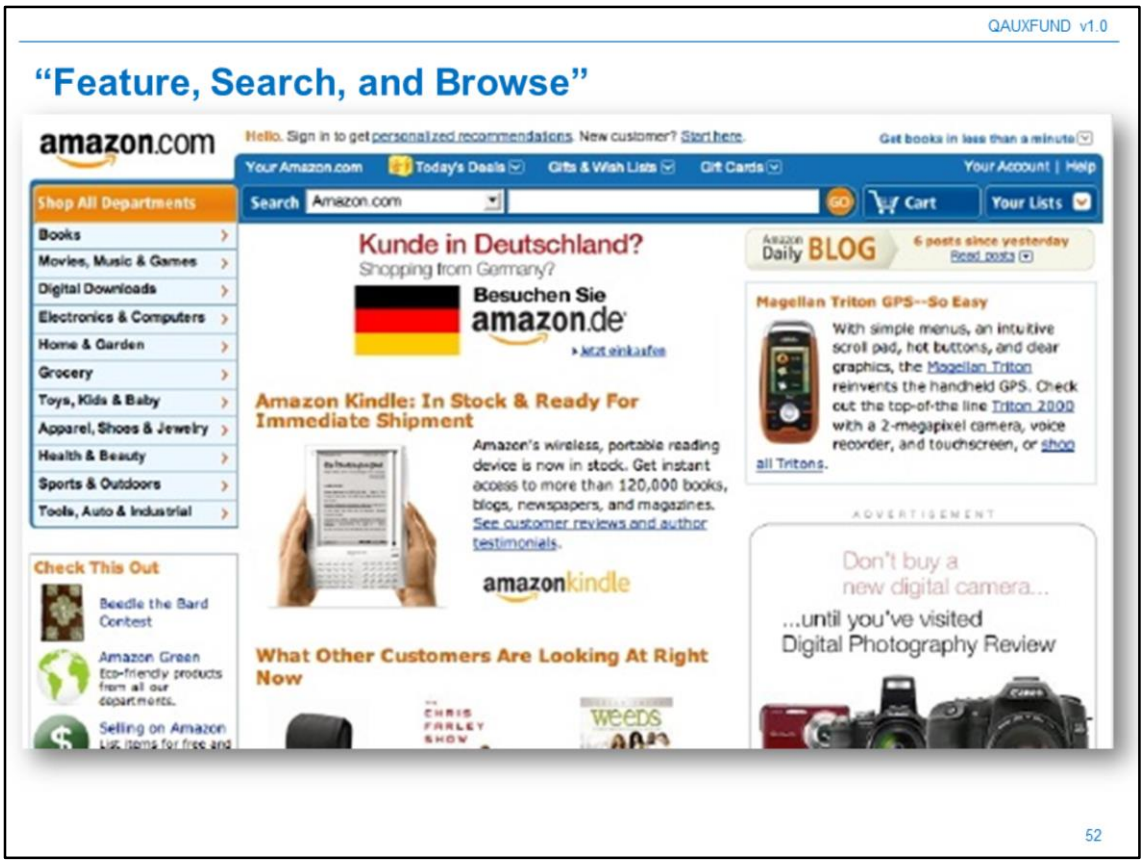
For more, read “Designing Interfaces” by Jenifer Tidwell

QAUXFUND v1.0

Some Common Patterns

- There are so many patterns out there and in common use that we can not hope to do more than touch on them
- Here are a selection of a few common examples as demonstrations of what a pattern is, how they are presented, and how they is used

51



What

Three elements on the main page of the site or app; a featured article or product, a search box, and a list of items or categories that can be browsed

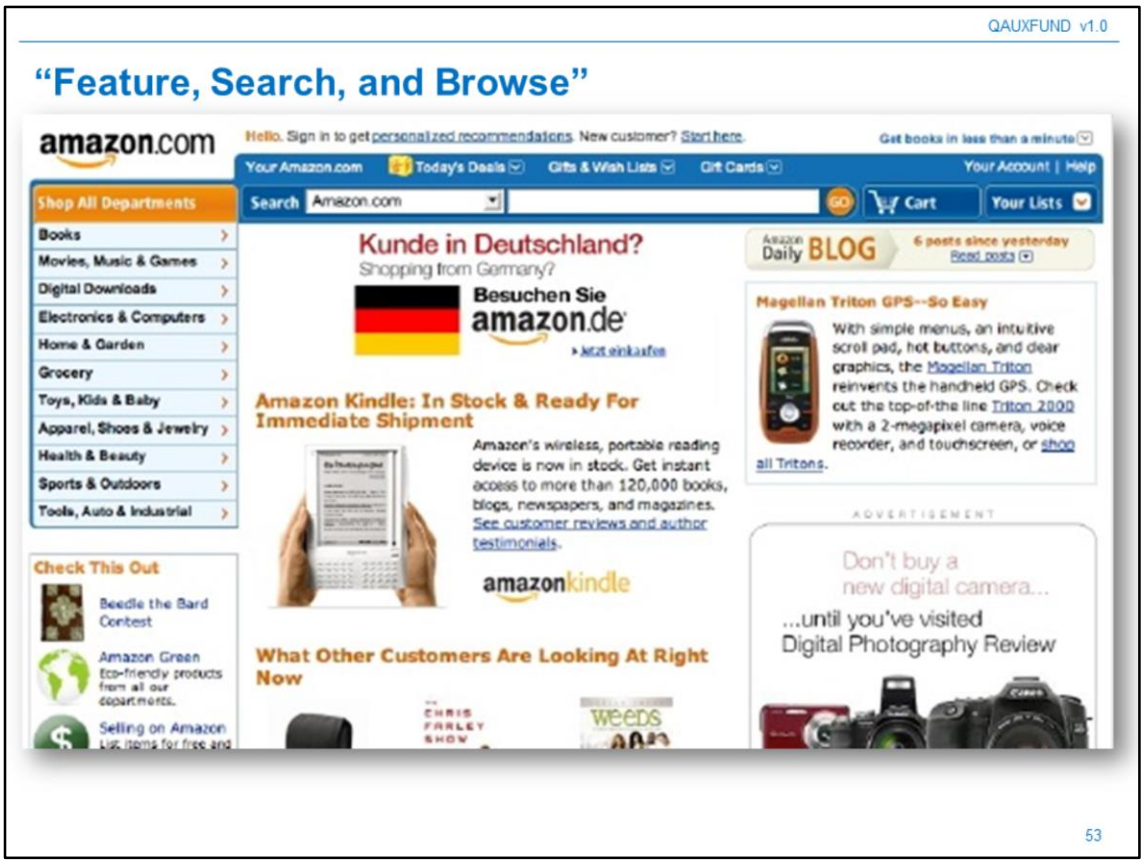
Use when

Your site offers a long list of items (articles, products, videos etc.) that can be browsed and searched. You want to engage incoming users immediately by giving them something interesting to read or watch

Why

Searching and browsing go hand in hand as two ways to find desired items. Some users will know what they are looking for and enter that into the search box, while others will do more open ended browsing through lists and categories

Featured items are there to "hook" users and provide your user with something to read or experiment with without having to do any additional work.



How

Place a search box in a prominent location and use whitespace to set it apart from it's surroundings.

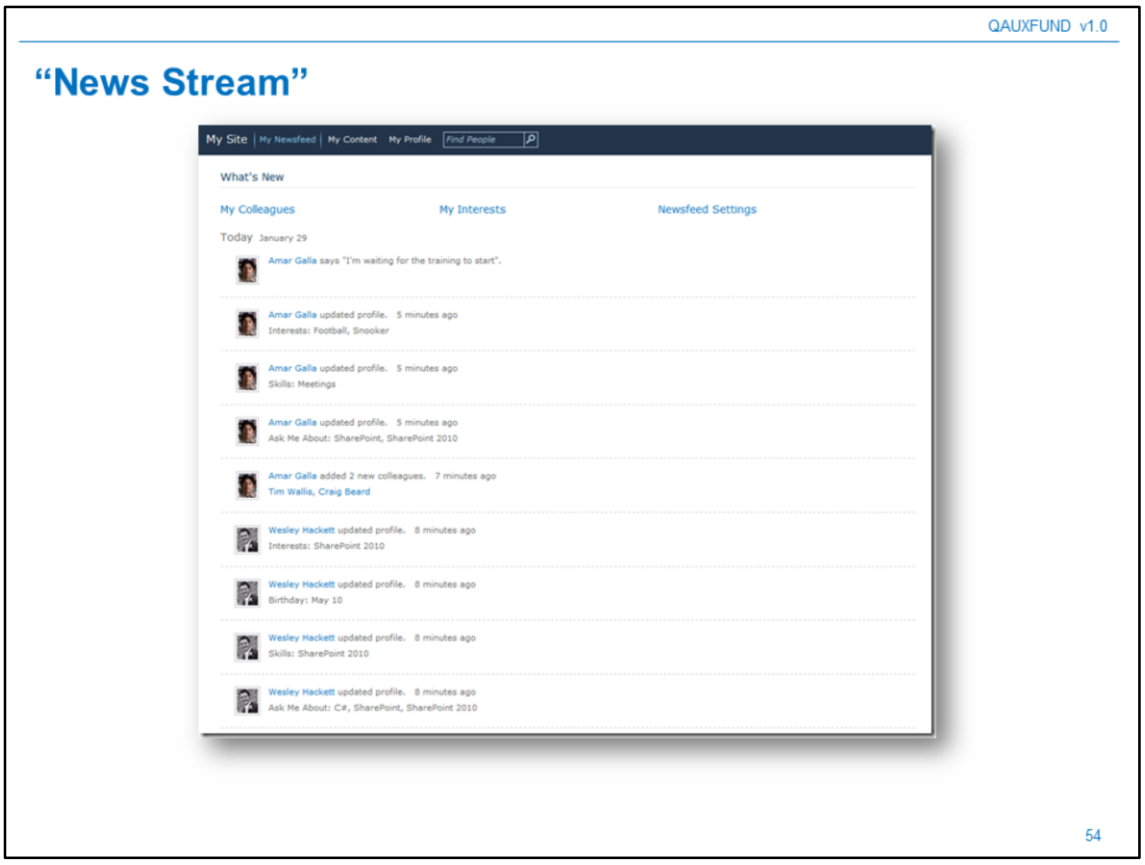
Try to eliminate all other text fields above the fold to make sure that users do not confuse those with the search box.

Set the featured item centre stage. Very near it (and still above the fold) place an area for browsing the rest of the site's content. This is usually a list of topics or product categories

Examples

Most news websites

Most ecommerce websites



What

Show time-sensitive items in a reverse chronological list, with the latest items at the top. Update it dynamically, and combine the items from different sources or people into one single stream.

Use when

Your site or app uses one or more communication channels, such as blogs, email, social site updates, or news sites, to deliver timely content to users.

This channel may be personal; a user "owns" it, like an email client or Facebook friends list, or public, such as a website or public Twitter stream.

Why

People can keep up with a news stream easily, since the latest items reliably appear on top with no effort on the part of the user. They can check in often and be assured of seeing what they need to see.

A glance at a News Stream application can give a user lots of useful information (or entertainment) with very little time or effort.

How

List incoming items in reverse chronological order. If the technology permits, “push” new items onto the top of the list without waiting for the user to request an update, but offer a way for the user to get an immediate update or refresh anyway.

Very busy streams can be split up into manageable sub streams by topic, sender, source, search terms, or other factors—you could let the user choose which one(s) to show. Services such as Facebook, FriendFeed, Twitter, and some RSS readers show clickable lists of these sub streams to the left or right of the incoming content. Others, such as Tweetdeck, use Many Workspaces to show multiple parallel panels of incoming content.

Information shown with each item might include:

- **What:** For short micro-updates, show the whole thing. Otherwise, show a title, a teaser that’s a few words or sentences long, and a thumbnail picture if one is available.

- **Who:** This might be the person who wrote an update, the blog where an article was posted, the author of said article, or the sender of an email. Actual person names humanize the interface, but balance this against recognition and authoritativeness—the names of news outlets, blogs, companies, and so forth are important, too. Use both if that makes sense.

- **When:** Give a date or timestamp; consider using relative times, such as “Yesterday” and “Eleven minutes ago.”

- **Where:** If an item’s source is a website, link to that website. If it comes from one of your organization’s blogs, link to that. (But here’s another interpretation of “where”: can you get geolocation data about the item, and show it on a map?)
- When there’s more to an item than can be shown easily in the list display, show a “More” link or button. You might design a way to show the entire contents of an item within the News Stream window.

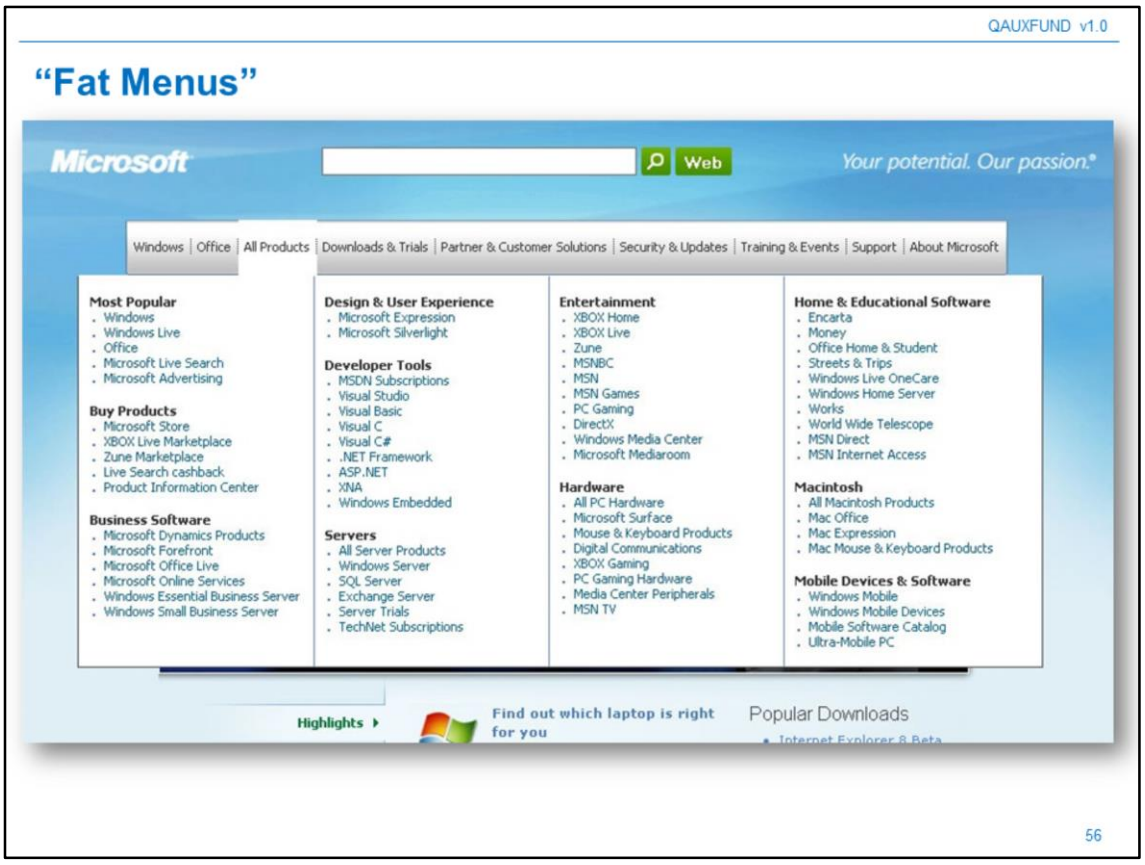
Examples

Reddit

Google News

Facebook

Twitter



What

Display a long list of navigation options in drop-down or fly-out menus. Use these to show all the subpages in site sections. Organize them with care, using well-chosen categories or a natural sorting order, and spread them out horizontally.

Use when

The site or app has many pages in many categories, possibly in a hierarchy with three or more levels. You want to expose most of these pages to people casually exploring the site, so they can see what's available. Your users are comfortable with drop-down menus (click to see them) or fly-outs (roll over them with the pointer).

Why

Fat Menus make a complex site more discoverable. They expose many more navigation options to visitors than they might otherwise find.

By showing so many links on every page, you make it possible for a user to jump directly from any subpage to any other subpage (for most subpages, anyhow). You thus turn a multi-level site (where subpages aren't linked to the subpages in other site sections) into a fully connected site.

Fat Menus are a form of progressive disclosure, an important concept in user interface design. Complexity is hidden until the user asks to see it. A visitor to a site that uses these can look over the menu headings to get a high-level idea of what's there, and when he's ready to dive in, he can open up a Fat Menu with a gesture. He isn't shown millions of subpages before he's ready to deal with them.

If you're already using menus in your global navigation, you might consider expanding them to Fat Menus if surfacing more links makes the content more attractive to casual browsers. People won't have to drill down into categories and subcategories of your site hierarchy in order to discover interesting pages; they'll see them there, right up front.

How

On each menu, present a well-organized list of links. Arrange them into Titled Sections if they fit into subcategories; if not, use a sorting order that suits the nature of the content, such as an alphabetical or time-based list.

Use headers, dividers, generous whitespace, modest graphic elements, and whatever else you need to visually organize those links. And take advantage of horizontal space—you can spread the menu across the entire page if you wish. Many sites make excellent use of multiple columns to present categories. If you make the menu too tall, it might go right off the end of the browser page. (The user controls how tall the browser is; guess conservatively.)

The best sites have Fat Menus that work stylistically with the rest of the site. Design them to fit well into the colour scheme, grid, and so on of the page.

Some menu implementations don't work well with accessibility technology such as screen readers. Ensure that your Fat Menus can work with these. If they can't, consider switching to a more static strategy, such as a Sitemap Footer.

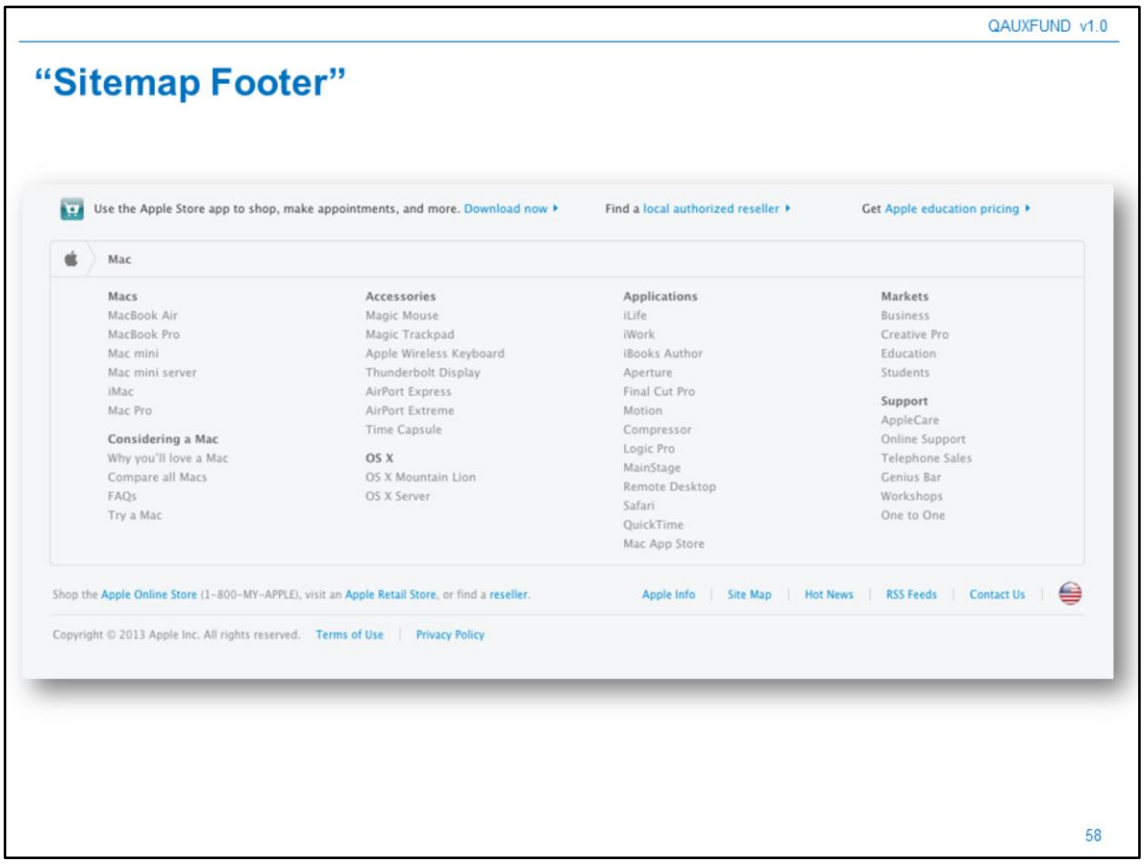
Examples

Microsoft.com

Slate.com

British Red Cross

Nike.com



What

Place a site map into the footer of every page in a site. Treat it as part of the global navigation, complementary to the header. Abridge the site map if you need to make it fit into a compact space.

Use when

The site you're designing uses a generous amount of space on each page, and you don't have severe constraints on page size or download time. You don't want to take up too much header or sidebar space with navigation.

The site has more than a handful of pages, but not an outrageously large number of categories and "important" pages (things that users will look for). You can fit a reasonably complete site map, at least for pages that aren't in the header, into a strip no taller than about half of a browser window.

There may be a global navigation menu in the page header, but it doesn't show all levels in the site hierarchy; maybe it only shows the top-level categories. You prefer a simple, well-laid-out footer instead of "Fat Menus", perhaps because of implementation ease or accessibility issues.

Why

Sitemap Footers make a complex site more discoverable. They expose many more navigation options to visitors than they might otherwise have.

By showing so many links on every page, you make it possible for a user to jump directly from any subpage to any other subpage (or major page, anyhow). You thus turn a multi- level site where subpages aren't linked to the subpages in other site sections, into a fully connected site. The footer is where the user's attention lands when she reads to the end of a page. By placing interesting links there, you entice the user to stay on the site and read more.

Finally, showing users the whole site map gives them a strong sense of how the site is constructed and where they might find relevant features. In complex sites, that could be valuable.

You may find yourself trying to choose between a Sitemap Footer design and a "Fat Menus" design. In conventional websites, a Sitemap Footer would be easier to implement and debug because it doesn't depend on anything dynamic: instead of showing fly-out menus when the user rolls over items or clicks on them, a Sitemap Footer is just a set of static links. It's also easier to use with screen readers and it doesn't require fine pointer control, so it wins on accessibility as well.

On the other hand, the footer may be ignored by busy or casual users who focus only on the page content and the headers. Usability-test if you have any doubts, and watch the click metrics to see if anyone even uses the Sitemap Footer.

How

Design a page-wide footer that contains the site's major sections (categories) and their most important subpages. Include utility navigation, tools such as language choice or Social Links, and other typical footer information such as copyright and privacy statements.

This might constitute a complete site map for your site, or it might not. The idea is to cover most of what visitors need to find, without overloading the header or sidebar navigation. In practice, what often happens is that the global navigation options at the top of the page reflect a more task-oriented design—it tries to answer visitors' immediate questions regarding "What is this about?" and "Where do I find X right this second?" Meanwhile, the Sitemap Footer shows the actual hierarchical structure of the site itself. This two-part arrangement appears to work well.

If your site deals with content that itself requires complex navigation—such as a large set of products, news articles, music, videos, books, and so on—you could use the top of the page for content navigation and the Sitemap Footer for almost everything else.

Here are some features that can often be found in Sitemap Footers:

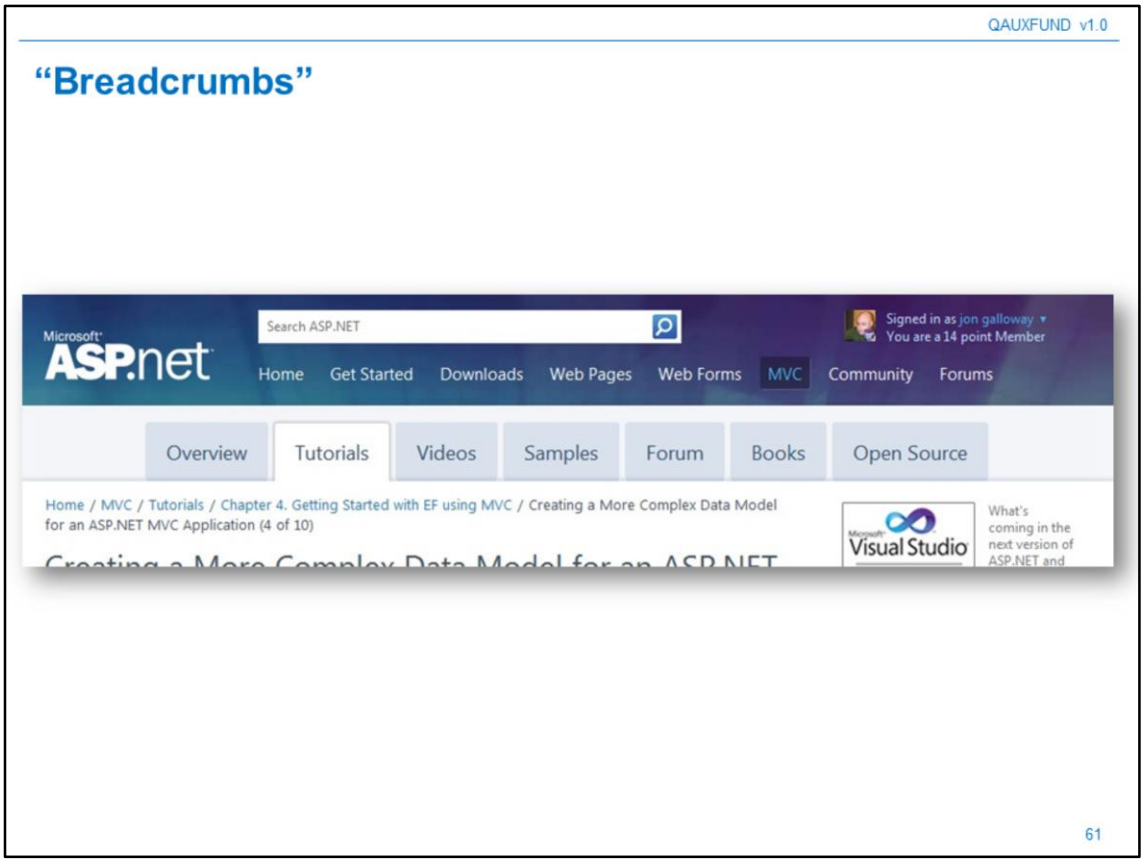
- Major content categories
- Information about the site or organization
- Partner or sister sites—for example, sites or brands owned by the same company
- Community links, such as forums
- Help and support
- Contact information
- Current promotions
- Donation or volunteer information, for non-profits

Examples

The Apple Store website

Microsoft

Weebly.com



What

On each page in a deep navigational hierarchy, show a list of all the parent pages up to the main or home page

Use when

Your application or site has a hierarchical structure with two or more levels.

Why

Breadcrumbs show each level or hierarchy leading to the current page from the top of the application all the way down. In a sense they show a single linear "slice" of the overall map of the site or app.

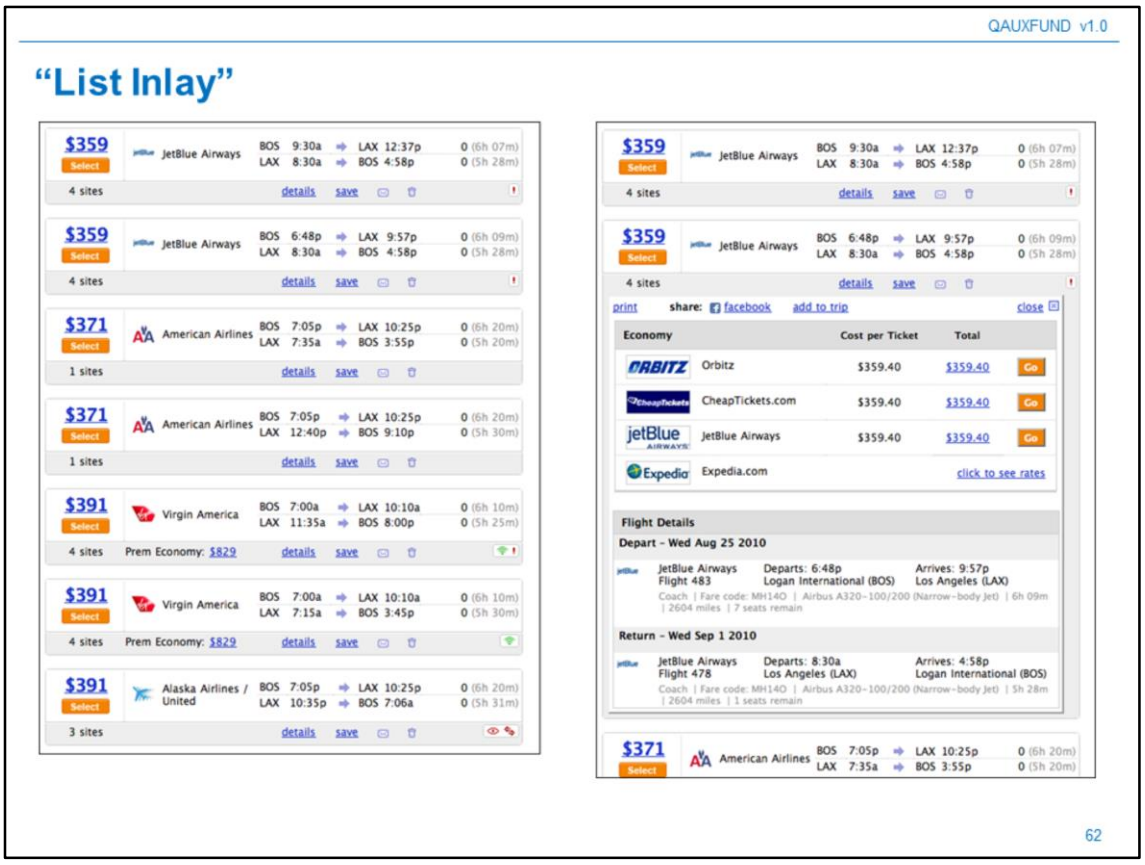
Breadcrumbs are usually clickable links that allow the user to use them as a navigational device.

How

Near the top of the page, put a line of text or icons indicating the current level of hierarchy. Start with the top level, and place the nest level to it's right, continuing on until the current level. Between the levels put a graphic or text character to indicate the parent/child relationship. This is usually an angle bracket, but can be a slash, or any other symbol.

Examples

The Windows 7 control panel



What

Show a list of items as rows in a column. When the user selects an item, open that item's details in place, within the list itself. Allow items to be opened and closed independently of each other.

Use when

You have a list of items to show. Each item has interesting content associated with it, such as the text of an email message, a long article, a full-size image, or details about a file's size or date. The item details don't take up a large amount of space, but they're not so small that you can fit them all in the list itself.

You want the user to see the overall structure of the list and keep that list in view all the time, but you also want her to browse through the items easily and quickly. Users may want to see two or more item contents at a time, for comparison.

The list of items has a vertically oriented, columnar structure.

Why

A List Inlay shows an item's details within the context of the list itself. The user can see the surrounding items, which might help in understanding and using the item contents.

Also, a user can see the details of multiple items at once. This is not possible in most other ways of displaying item details. If your use cases call for frequent comparison of two or more items, this might be the best option.

How

Show list items in a column. When the user clicks on one, open the item in place to show the details of that item. A similar gesture should close the item back up again.

When an item is opened, enlarge the item's space downward, pushing the subsequent items down the page. Other items do the same when opened. A scrolled area should be used to contain this ever-changing vertical structure, since it could get very tall indeed!

To close the details panel, use a control that clearly indicates its purpose (e.g., "Close" or "X"). Some implementations of List Inlay only put that control at the end of the details panel, but users may need it at the top if the panel is long and they don't want to move down the whole thing. Put a closing control very near the original "open" control (or replace one with the other). This at least ensures that the user's pointer won't move very far if she wants to open an item, glance at it, close it, and move on.

Use an Animated Transition as the item opens and closes, to keep the user oriented and to focus attention on the newly opened item.

If your application permits the user to edit items, you could use a List Inlay to open an editor instead of item details (or in addition to them).

A list that uses List Inlays works the same way as an Accordion: everything lies in a single column, with panels opening and closing in situ within it.

Examples

Google Reader uses a List Inlay within the context of a Two-Panel Selector. It has a multi-level hierarchy of containers to present; the containers are shown in the tree selector on the left, but the list of articles takes up Centre Stage and the user can then open them in place to read them.

What

Arrange content items in a grid or matrix. Each item should follow a common template, and each item's visual weight should be similar. Link to jump pages as necessary.

Use when

The page contains many content items that have similar style and importance, such as news articles, blog posts, products, or subject areas. You want to present the viewer with rich opportunities to preview and select these items.

Why

A grid that gives each item equal space announces that they have equal importance. The common template for items within the grid tells the user that the items are similar to each other. Together, these techniques establish a powerful visual hierarchy that should match the semantics of your content.

Grids look neat, ordered, and calming. That may suit the style of your site or app.

How

Figure out how to lay out each item in the grid. Do they have thumbnail images or graphics? Headlines, subheads, summary text? Links to jump pages (e.g., a page with the full story)? Render them with more than just blocks of body text: make headlines of different colours, be creative with whitespace, and use images if you can do so evenly across all items. Experiment with ways to fit all the right information into a relatively small space— tall, wide, or square—and apply that template to the items you need to display.

Now arrange the items in a grid. You could use a single row, or a matrix that's two, three, or more items wide. Consider page width as you do this design work—what will your design look like in a narrow window? Will most of your users have large browser windows? What happens on tiny mobile devices?

You may choose to highlight grid items, either statically (to emphasize one item over others) or dynamically, as a user hovers over those grid items. Use colour and other stylistic changes, but don't change the positions, sizes, or other structural elements of the grid items—you don't want content jumping around as the user hovers over different items!

Examples

MapQuest Travel blogs

Amazon uses lots of single row grids in a "Grid of Equals" pattern on it's home page.

What

Put two side-by-side panels on the interface. In the first one, show a list of items that the user can select at will; in the second one, show the content of the selected item.

Use when

You have a list of items to show. Each item has interesting content associated with it, such as the text of an email message, a long article, a full-sized image, contained items (if the list is a set of categories or folders), or details about a file's size or date.

You want the user to see the overall structure of the list and keep that list in view all the time, but you also want him to be able to browse through the items easily and quickly. People won't need to see the details or content of more than one item at a time.

Physically, the display you're working with is large enough to show two separate panels at once. Very small cell phone displays cannot cope with this pattern, but many larger mobile devices can.

Why

This is a learned convention, but it's an extremely common and powerful one. People quickly learn that they're supposed to select an item in one panel to see its contents in the other. They might learn it from their email clients, or from Windows Explorer, or from websites; whatever the case, they apply the concept to other applications that look similar.

When both panels are visible side by side, users can quickly shift their attention back and forth, looking now at the overall structure of the list ("How many more unread email messages do I have?"), and now at an object's details ("What does this email say?"). This tight integration has several advantages over other physical structures as it reduces physical effort. The user's eyes don't have to travel a long distance between the panels, and he can change the selection with a single mouse click or key press rather than first navigating between windows or pages (which can take an extra mouse click).

It reduces visual cognitive load. When a window pops to the top, or when a page's contents are completely changed the user suddenly has to pay more attention to what they are now looking at; when the window stays mostly stable, the user can focus on the smaller area that did change. There is no major "context switch" on the page.

It reduces the user's memory burden. Think about the email example again: when the user is looking at just the text of an email message, there's nothing on-screen to remind him of where that message is in the context of his inbox. If he wants to know, he has to remember, or navigate back to the list. But if the list is already on-screen, he merely has to look, not remember. The list thus serves as a "You are here" signpost.

How

Place the selectable list on the top or left panel, and the details panel below it or to its right. This takes advantage of the visual flow that most users who read left-to-right languages will expect (so try reversing it for right-to-left language readers).

When the user selects an item, immediately show its contents or details in the second panel. Selection should be done with a single click. But while you're at it, give the user a way to change his selection from the keyboard, particularly with the arrow keys—this helps reduce both the physical effort and the time required for browsing, and contributes to keyboard-only usability.

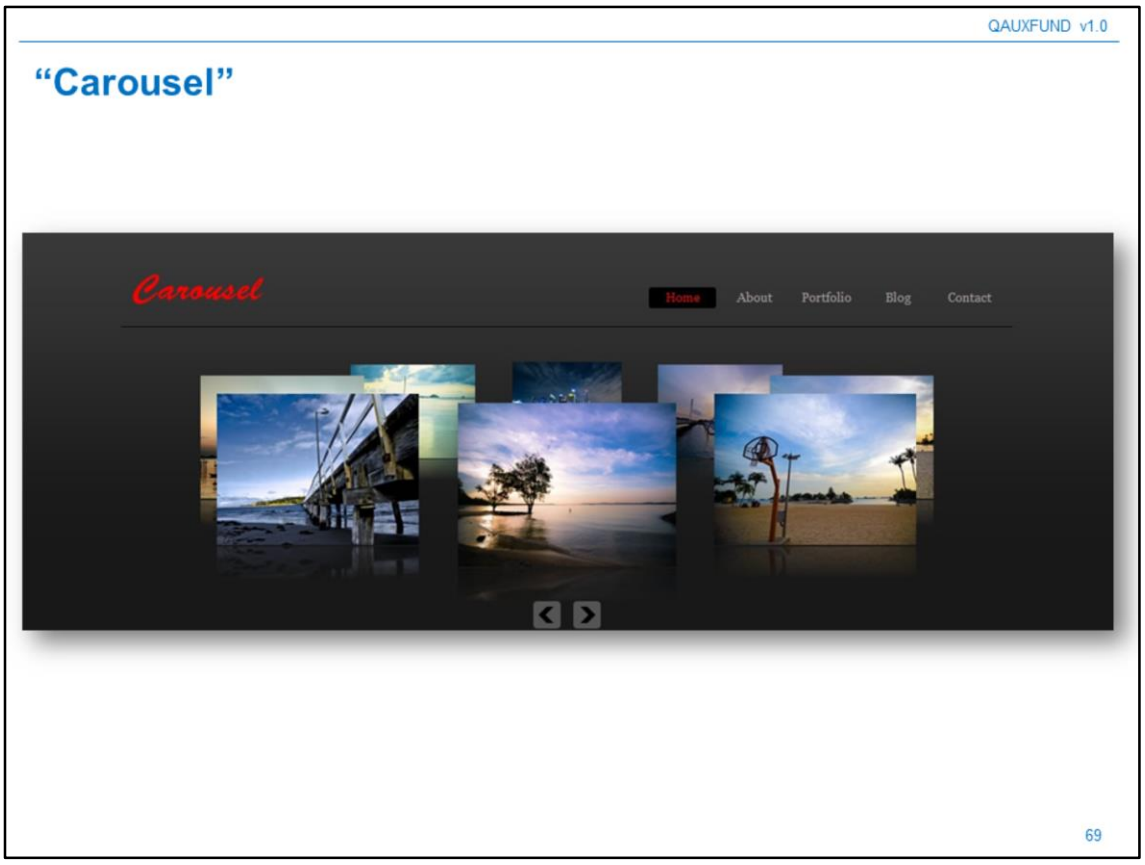
Make the selected item visually obvious. Most GUI toolkits have a particular way of showing selection (e.g., reversing the foreground and background of the selected list item). If that doesn't look good, or if you're not using a GUI toolkit with this feature, try to make the selected item a different colour and brightness than the unselected ones—that helps it stand out.

What should the selectable list look like? It depends—on the inherent structure of the content, or perhaps on the task to be performed. For instance, most file system viewers show the directory hierarchy, since that's how file systems are structured. Animation and video editing software use interactive timelines. A GUI builder may simply use the layout canvas itself; selected objects on it then show their properties in a property editor next to the canvas.

When the select-and-show concept is extended through multiple panels to facilitate navigation through a hierarchical information architecture, you get the Cascading Lists pattern.

Examples

Outlook



What

Arrange a list of visually interesting items into a horizontal strip or arc and let the user scroll or swipe the thumbnails back and forth to view them. Enlarge the centre item if appropriate

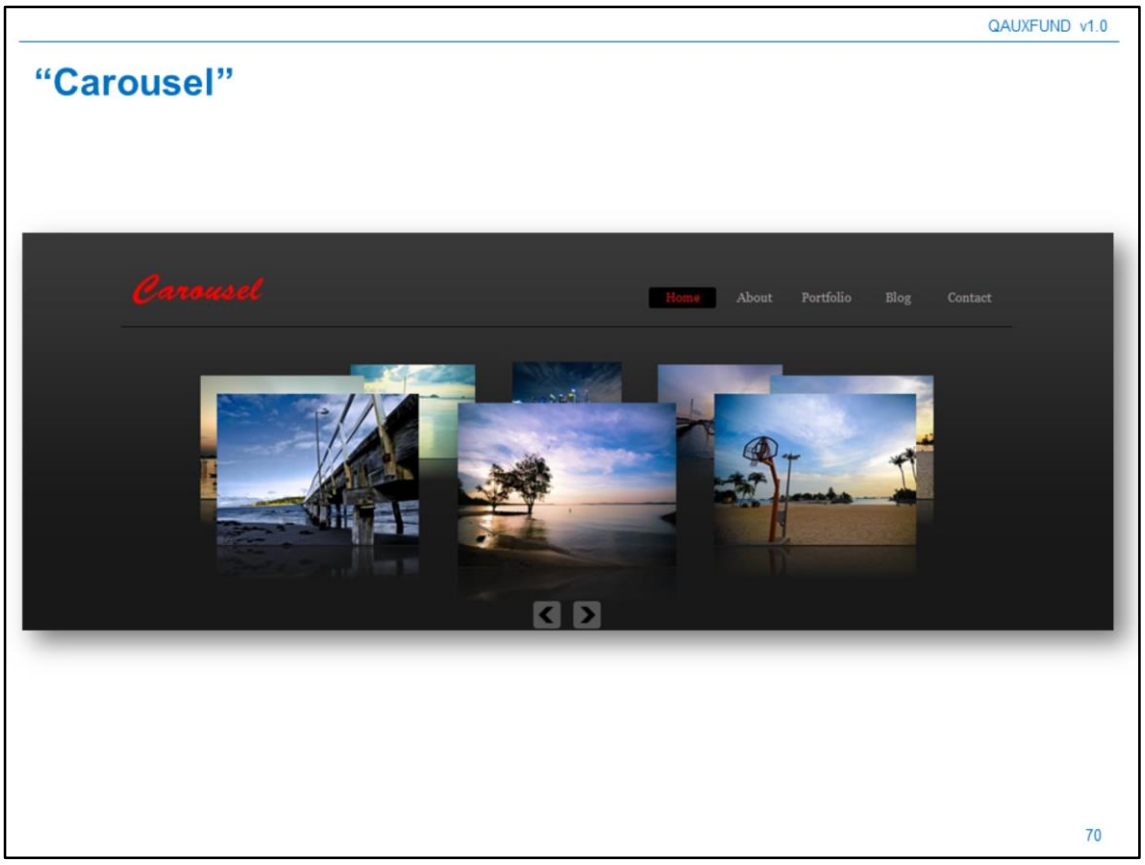
Use when

The list items have visual representations that uniquely identify them. E.g. logos, screen captures, album art etc..

The list is flat; i.e. it is not divided into categories

Why

A carousel offers an engaging interface for browsing visual items, encouraging the user to inspect the items that are in view, and to see what is next. A user can not easily jump to a certain point deep in the list so has to scroll through everything. This pattern encourages browsing.



How

Create thumbnails for each item showing in the carousel. Place the text metadata close to the thumbnail, but in small print in order to maintain the thumbnail's visual prominence.

Arrange the thumbnails horizontally and show a small number of them on either side. Those that are not shown, are hidden. Provide large arrows for paging left and right through the carousel

Examples

Amazon provides suggested products in a carousel

Cover Flow in iTunes

Exercise – Making Interaction Choices

- **Choose one of your Task Analysis outcomes**
  - Examine how you may choose to build a widget for this
    - Decide on the structure of your base items (e.g. buttons, links)
    - Explore the task and sketch out a solution on a widget sheet
    - Layout a description of the interaction

Stopping Points

- Whenever the user is stopped from continuing down their course of action, that is a stopping point
- Sometimes it is important to create a stopping point, but more often than not they are less desirable
- Ensure not to stop the proceedings with unnecessary interruptions
- Where appropriate, allow the user to permanently dismiss a dialogue so that the application does not ask the user that question in the future

72

Errors

- All too often the process is halted by an error message
- If that error message has no suggestions for a next course of action then the design is wrong
- If there is nothing that the user can do, as a designer you should consider the existence of the error message to be a bug

73

Notifications

- Notification messages should never stop proceedings
- If your product ever has a notification that merely requires the user to click on an “OK” button, but has no other options, then the design is faulty

74

Confirmations

- "Are you sure you want to quit?"
- "Do you want to leave this page?"

There are times when it is important to confirm a request, and that is when an irreversible and costly mistake is potentially being made

Never ask a user to make a potentially irreversible and costly decision in a confirmation dialogue; the information that they will need in order to make that decision almost certainly cannot be displayed in the dialogue, making it the wrong method for parsing the decision

75

Exercise – Reviewing User Interactions

- **Return to the widgets you created earlier in this module**
  - With the information we have uncovered since create a strategy for:
    - Stopping points
    - Notifications
    - Warnings
  - Review designed interactions and consider whether they uphold Fitt's law and are good human processors


QAUXFUND v1.0
Content Strategy
transforming performance
through learning

Content

- **One of the first steps to effective IA is content analysis determine:**
  - Content you have
    - Learn what you have, avoid duplication
  - Content you need
    - For each task analyse content required, who is responsible for it
    - Who will provide content
  - Communicating about content
    - Build a content spreadsheet
    - Colour code according to popularity
  - Content planning
  - Classification schemes

78

Content you have

- **Learning what content you have will help you:**
  - Understand the subject
    - If you're working on a new site, or a subject you don't have much experience with
  - Look at it with fresh eyes:
    - A great refresher and gives you a chance to look at your content
  - Clean up:
    - Clean up old, out-of-date and inaccurate content.
  - Migrate it:
    - You need to know what you have so you don't lose anything in the migration.
  - Manage progress:
    - Keep track of things like what content has been rewritten

79

QAUXFUND v1.0

Conduct a content inventory

- **To manage content you will need to perform some form of audit:**
  - Full content inventory – which includes all
    - Page, downloadable objects e,g, PDF files, documents,
    - Embedded objects, such as video or audio other content
      - In blogging sites this may include reader comments
  - Partial content inventory
    - This collects information for a number of levels of the site
      - Though it may show some parts of the site in more detail.
    - It is a good practical approach for very large sites
  - Content audit
    - Is a sample across the site.
    - It may collect details for the first couple of levels of a site
      - Then details for different page types in each section.

80

QAUXFUND v1.0

Content you Need

- **Whether working on a new design or launching think content**
  - The content you choose should meet
    - The needs of the people using your product
    - Help achieve your project goals
- **Consider approaching content creation in the following ways:**
  - Current behaviours
  - User research
  - Your own ideas for content

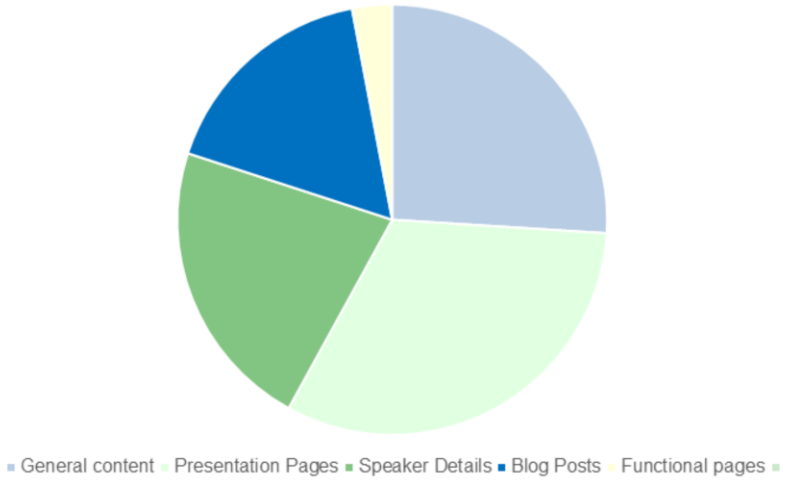
81

Current Behaviour

- **Examine what users are currently viewing and doing**
  - From site metrics
  - Competitor analysis
- **Do not just focus on immediate data**
  - Check what happens at different times of the year
  - When new content is added – does it peak and not get used again?
- **Prioritise popular content – it becomes popular because of users**
  - Provide better links to existing and related content
  - Investigate social networking trends
    - Use verbs and action from the research phase

Content Planning

- **Build a spreadsheet of content**
  - Use analytics to investigate who is doing what
  - Categorise content
  - Understand popularity



QAUXFUND v1.0

User Journeys

User Journeys detail the exact steps required to complete a task

- **They differ from Task Models**
  - They show required interactions through a system
  - Rather than being a representation of user behaviour
  - Show specific routes through a site
  - Rather than logical structure of entire site

- **A user journey reflects the behaviour uncovered in the task model**
  - Ensuring the task model and user journey line up helps verify analysis

84

QAUXFUND v1.0

When to Use User Journeys

Developed once you have solid personas and task models

- **User journeys should be created in the following scenarios**
  - Product Development:
    - Create a user journey when you create a system from scratch
    - They provide the framework to build the interface
      - Align to wireframes
  - Analysis:
    - Useful documents to help with testing
      - Step back from page by page to the entire journey
      - Show journeys and describe the positives and pain points
      - Show how the existing journey needs improvement

85

QAUXFUND v1.0

What User Journeys Communicate (1)

User Journeys should show the following:

- **The Goal/Task**
  - Use the goal as a page title
  - Add the desired outcome *e.g. buy the items in the shopping cart*
- **Steps**
  - Use flowcharts to demonstrate specific steps to complete the journey
    - e.g review a piece of data prior to submission
- **Decision Points:**
  - Show where a user has to make a choice in order to continue
    - e.g. Choose a cinema or film
- **Start and end steps:**
  - The beginning and the end of a journey
  - Use different shaped blocks to help them stand out

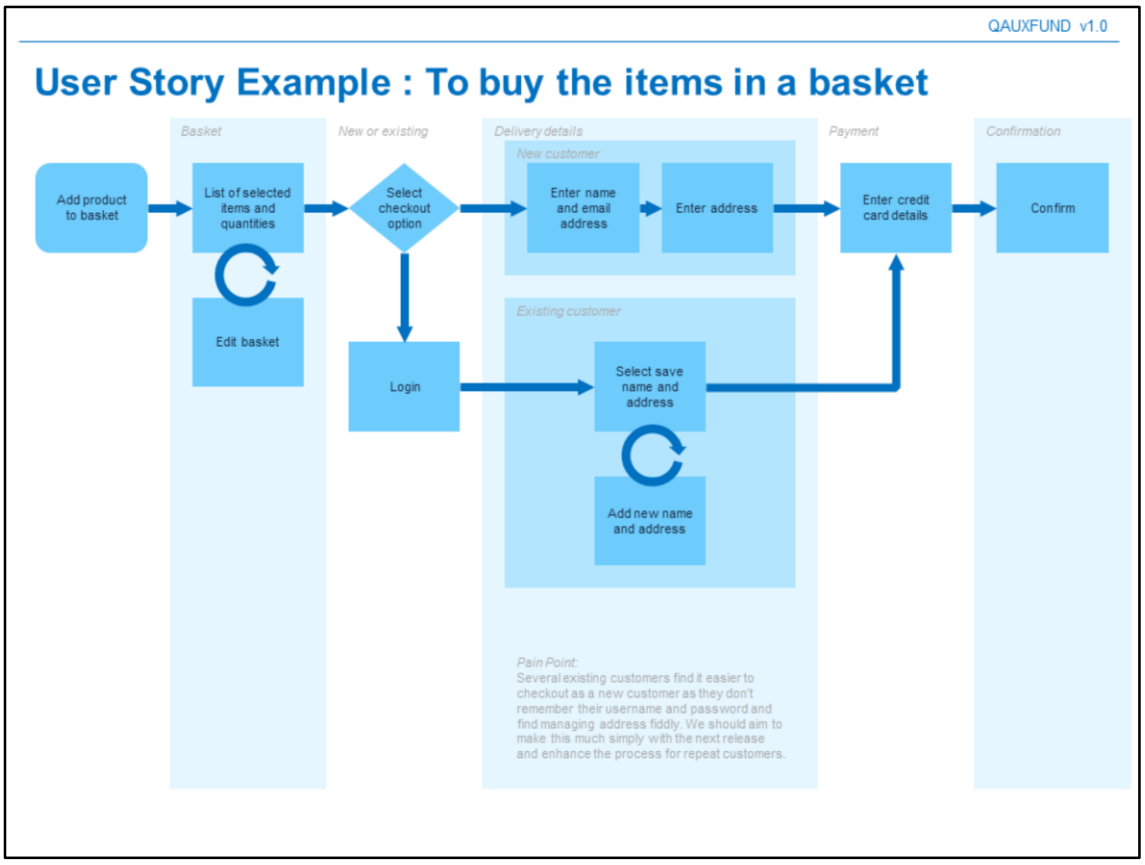
86

QAUXFUND v1.0

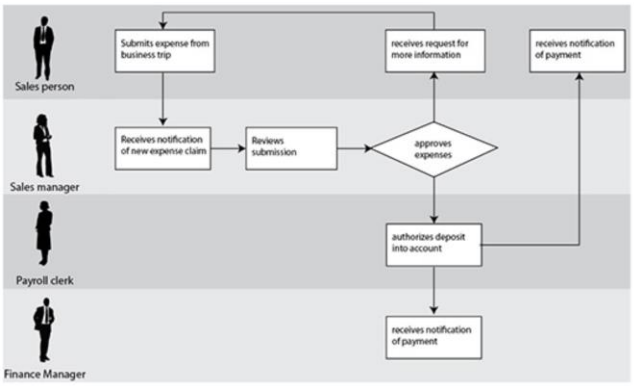
What User Journeys Communicate (2)

- **Grouping**
  - Where steps relate to each other grouping should be shown
  - Ensuring they make it to the correct wireframes
- **Flow:**
  - Single and double headed arrows show the user direction
  - Curved arrows can be an effective way of showing task repetition
    - e.g. Editing a cart more than once
- **Content:**
  - Can be used for any essential content requirement
    - e.g. a summary of an order
- **Pain points:**
  - Illustrate clear issues and problems in user tasks
  - Focus on fixing them

87


QAUXFUND v1.0
User Story Example : To buy the items in a basket
Basket
New or existing
Delivery details
New customer
Payment
Confirmation
Add product to basket
List of selected items and quantities
Edit basket
Select checkout option
Enter name and email address
Enter address
Enter credit card details
Confirm
Existing customer
Login
Select save name and address
Add new name and address
Pain Point:
Several existing customers find it easier to checkout as a new customer as they don't remember their username and password and find managing address fiddly. We should aim to make this much simply with the next release and enhance the process for repeat customers.

Workflow Diagrams - Swimlanes



http://www.lukew.com/ff/entry.asp?745

Review

- **The Structure phase gives us the building blocks for our physical design**
  - We need to develop navigation, tasks in a UCD way
  - We need to build in a consistent metaphor
  - We need to build consistent notifications, errors, stopping points
  - We need to build products users want to use
    - That pleases them to use
    - That may be because of need but its not an excuse for poor UCD