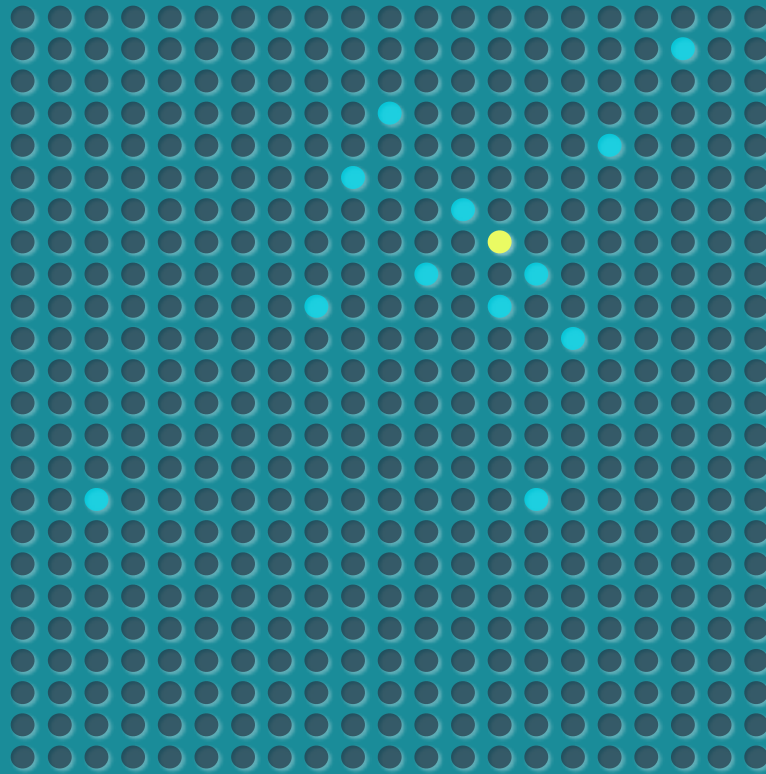


Bright ideas for



USER EXPERIENCE DESIGNERS

USERFOCUS



Dr David Travis @userfocus

THANKS FOR BEING PART OF THE USERFOCUS COMMUNITY

I love creating articles, resources and eBooks for the user experience community.

But it's my consulting and training work that pays the bills.

If you want to support the work that I do, here are some ideas.

ATTEND A WORKSHOP

Attend a public training courses. You'll find an up-to-date schedule here:

<https://www.userfocus.co.uk/training/index.html>

INVITE ME IN

I can bring my workshops to you and run them for a fixed all-inclusive fee no matter where you are. You'll find more information on in-house training here:

<https://www.userfocus.co.uk/training/in-house-training.html>

BUY AN ONLINE COURSE

Buy one of my online video courses on Udemy. You'll find a list here:

<https://www.udemy.com/user/davidtravis/>

COLLABORATE

Commission me to optimise the user experience of your product or service. You'll find more information on my consultancy services here:

<https://www.userfocus.co.uk/consultancy/index.html>

About this eBook	3
A CRAP way to improve usability	4
7 myths about paper prototyping	14
Help! What The Beatles can teach us about writing support material	26
Confessions of an Axure Master: 5 shortcuts for laying out Axure pages in record time	37
Communicating errors	43
4 ways to prototype faster	56
Why you need a user experience vision (and how to create and publicise it)	60
Card Games for Information Architects	69
Tips for writing user manuals	78
Five kinds of 'alt' text	85
About the authors	97
More eBooks from Userfocus	99

About this eBook

This is the second in our “Bright ideas” series of eBooks where we curate previously published articles from our web site. This edition focuses on articles of interest to User Experience Designers.

The articles in this collection were published on our web site between June 2007 and June 2012.

A quick word about the layout of this eBook. We have published this collection simultaneously as an ePub and as a PDF. Because our layout is aimed at making online reading an enjoyable experience, we’ve used a large font. But we’ve also made sure that printing remains a reasonable option. If you prefer to read a printed version, you’ll save paper and still find it very readable if you print this book as two pages per sheet.

Finally, if you enjoy this book and you’d like to hear about new collections the moment they’re published, sign up to receive our monthly newsletter by clicking on the link below.

— *David Travis*

[Get the Userfocus newsletter \(it’s free!\).](#)

A CRAP way to improve usability

David Travis

Visual design is often dismissed as eye candy. In fact, we can use four key principles of visual design to create more usable interfaces. These principles are Contrast, Repetition, Alignment and Proximity.

Within the field of user experience, visual design is sometimes perceived as a bit of an outsider. Indeed, some visual designers are apparently feeling a pressure to rebrand themselves as “user experience” designers to progress their careers. In my experience with clients, visual design is often likened to decorating the walls of your living room. It makes everything a little nicer, but it’s hardly a key part of the architecture of your house.

In fact, visual design has an important impact on the usability of designs. Research shows that people believe that more attractive designs are easier to use than less attractive designs — even when they’re not. This is known in psychology as the aesthetic usability effect and in folklore more simply as, “first impressions count”.

But good visual design offers more than improving people’s attitudes to a design. Good visual design will actually make interfaces easier to use.

There are at least four key principles of visual design that have an important impact on usability. These four principles —

contrast, repetition, alignment and proximity — were originally given the engaging acronym CRAP by Robin Williams (the visual designer, not the comedian). You can exploit these four principles to make user interfaces both more attractive and easier to use.

Let's look at each one in turn.

Contrast

Contrast in visual design helps to direct the viewer's eyes to what's important and helps them focus on what to do next. This means making your call to action — or whatever the focus of attention is meant to be on the page — very different from the other items that surround it.

In this example (Figure 1), the “Add to Basket” button is formatted identically to the “Cancel” button. This means that the user will have to carefully read the text on each button before making a choice.



Figure 1: Two buttons on a form, one labeled “Add to Basket” and the other labeled “Cancel”. Because the two buttons are formatted identically, it's not self-evident which is the primary and which is the secondary action.

With good use of contrast, we can make the default choice immediately apparent to users. For example, we can:

- Reduce the apparent depth of the button.

- Use a Roman (as opposed to a Bold) font.
- Remove the button entirely and change it to a hyperlink.

Figure 2 shows the impact of making these changes and with each change it becomes more self-evident which choice we expect the user to make.

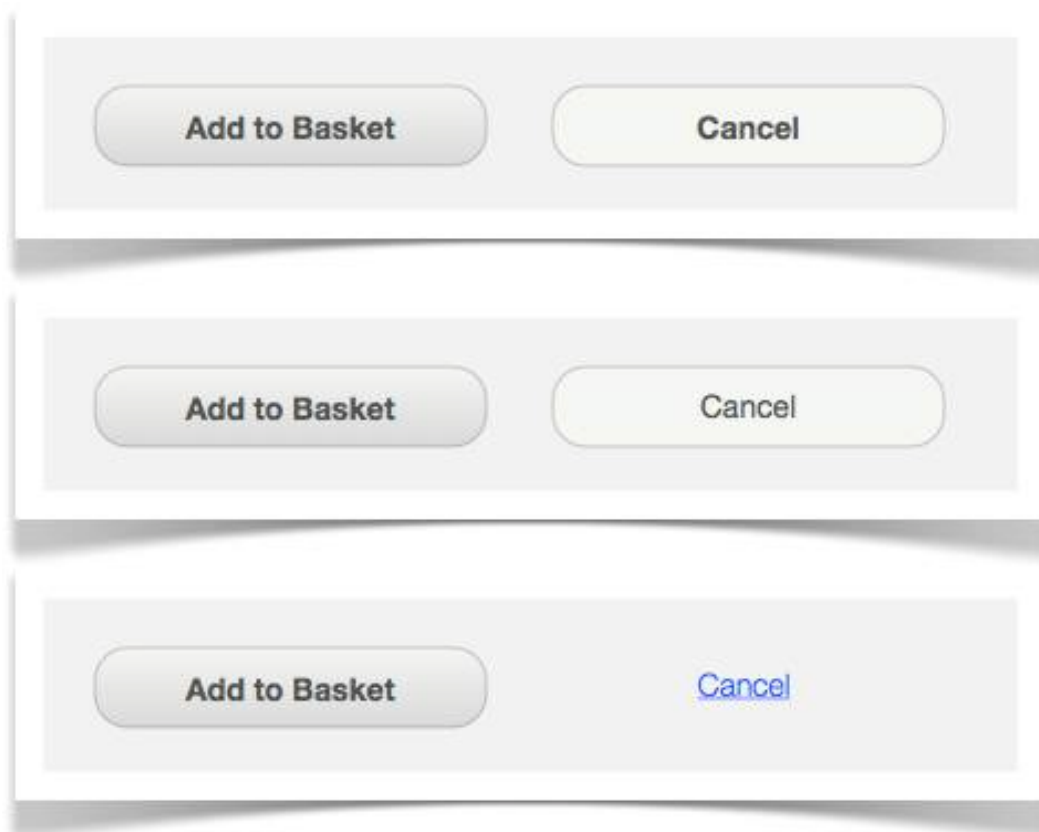


Figure 2: Three examples of using contrast to distinguish the primary action from the secondary action. The first example removed the apparent depth of the “Cancel” button; the second example builds on this and changes the font; and in the third example, the button is replaced with a hyperlink.

This example shows that you have many simple ways of creating contrast in your user interface, even if you think your design skills are limited. Simple text changes like bold, italic, underlining, uppercase, colour and highlighting may often be sufficient (although you obviously need to use these in moderation).

Contrast can also be used for more nefarious purposes. Figure 3 shows a voting paper used during the Anschluss (the 1938 annexation of Austria into Greater Germany by the Nazis). Note how the designer of this ballot paper has used contrast to make the 'default' choice ("Ja") easier for voters to identify. (Officially, 99.73% of people voted 'Yes' in this ballot.)

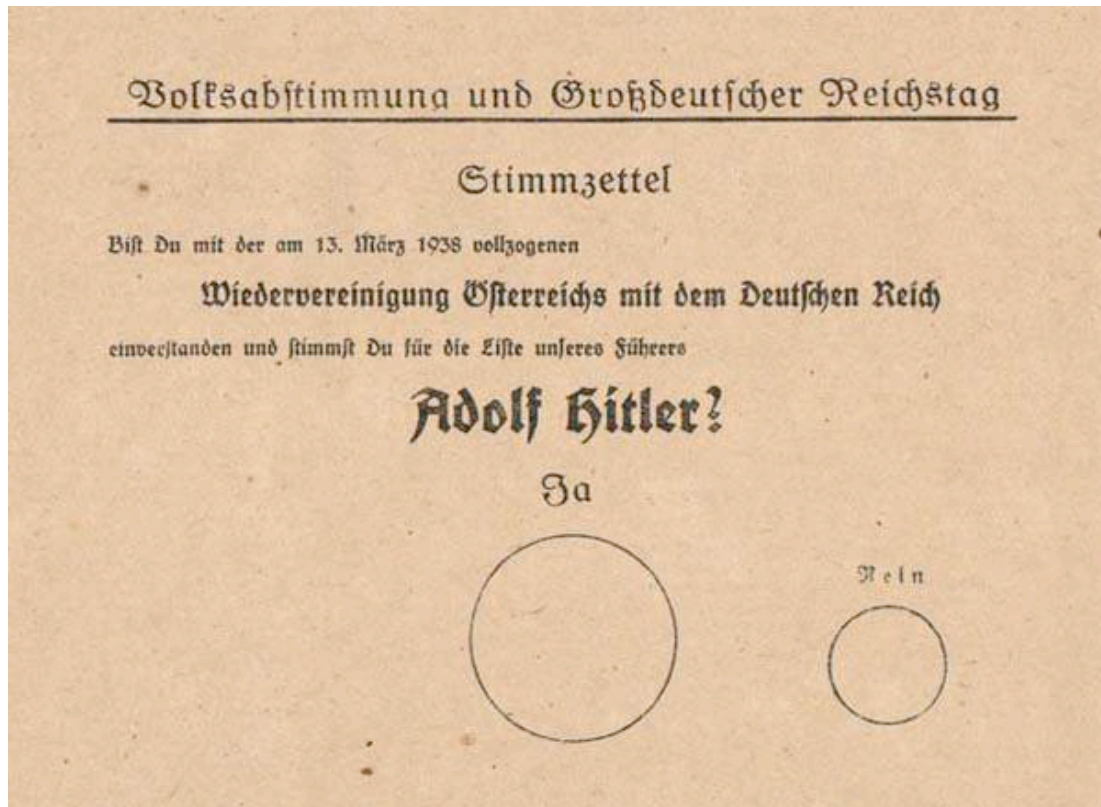


Figure 3: Voting paper from 10 April 1938. The text reads "Do you agree with the reunification of Austria with the German Empire that was enacted on 13 March 1938, and do you vote for the party of our leader Adolf Hitler?" The large circle is labeled "Yes" and the small circle is labeled "No". From Wikipedia.

Repetition

Repetition is really about consistency. (Sadly, 'Consistency' doesn't make a memorable acronym when combined with the other principles, so we're making do with 'Repetition' instead). At one level, this principle simply states that systems are more

usable (and easier to learn) when similar stuff is presented in similar ways. Note that this means both internal consistency (within your application) and external consistency (with the platform you're designing for). For a web site, internal consistency means making sure you use the same kind of fonts, icons, headings, links, list styles and page layout. External consistency means things like using standard buttons, link colours and search results. Jakob Nielsen has even coined a 'Law' for this: "Users spend most of their time on other web sites." People's expectations are formed by what they experience on other sites so if you do it differently your site will be harder to use. So ensure that people can find the common elements of a web page (such as page titles, site navigation, page navigation, search etc.) in the standard locations.

One of the best ways to use this principle intelligently is to think about the task your users are carrying out. Consider this example: if I asked you to put away the knives, forks and spoons in my kitchen, you would look for the cutlery tray. But if I then asked you to put away a penknife, you would probably look for the overstuffed drawer that contains items like the corkscrew and the tin opener. And if I then asked you to put away a box cutter, you would probably place it in the garage with the hammer, screwdrivers and other tools. This seems inconsistent: they are all knives, so why put them in different locations?

There's a deeper reason why consistency matters in user interface design and this example illustrates it. Your behaviour in these examples is entirely consistent with the task that you use these objects for. A knife is more like a fork than a box knife when the task is eating; but it is more like a hammer when the task is DIY. The task is the common denominator.

So when thinking about how to apply repetition in your user interface, think about the tasks that users carry out and how you can support that with your visual design.

Alignment

Alignment simply means making sure that all elements of the design line up horizontally and vertically. This can best be achieved by designing the interface to an underlying grid.

Alignment is probably the most dramatic visual treatment you can do to a design to make it appear visually easier to use. Take a look at Figure 4, below. Without thinking about it too hard, which form looks easier to complete, the one on the left or the one on the right?

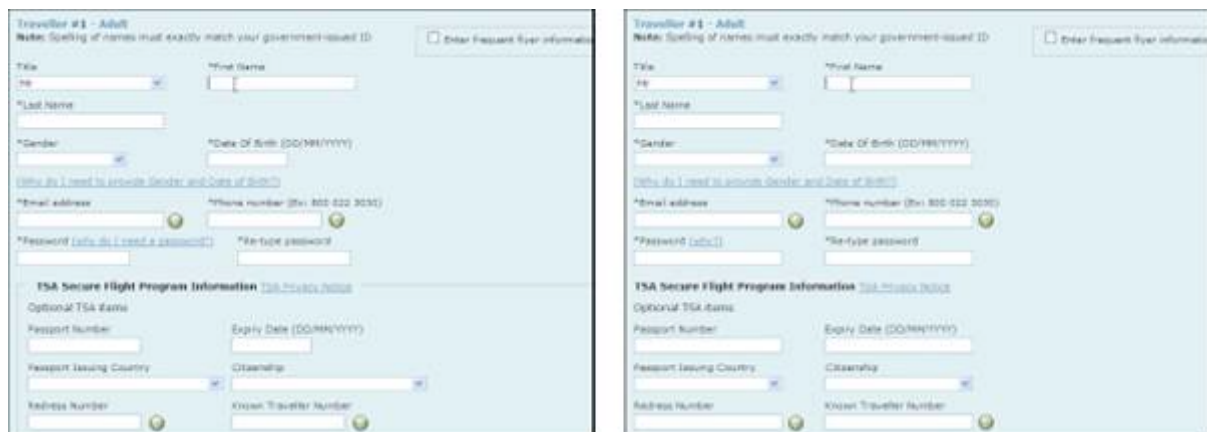
The image shows two side-by-side screenshots of a web form titled "Traveler #1 - Adult". Both forms contain the same fields: Title, First Name, Last Name, Gender, Date of Birth, Email address, Phone number, Password, TSA Secure Flight Program Information, Passport Number, Expiry Date, Passport Issuing Country, Citizenship, Address Number, and Known Traveler Number. The form on the left has fields that are not aligned with each other, creating a cluttered and difficult-to-read appearance. The form on the right has all fields aligned horizontally and vertically, creating a clean and organized appearance.

Figure 4: Two identical forms that differ only in their use of alignment.

Most people say the form on the right looks easier to complete. The two forms are identical except that the form fields have been properly aligned in the example on the right. In Figure 5, I have overlaid each form with red lines to show the various alignment lines. You can see that the form on the left, which most people describe as looking more complicated, also has more vertical alignment lines.

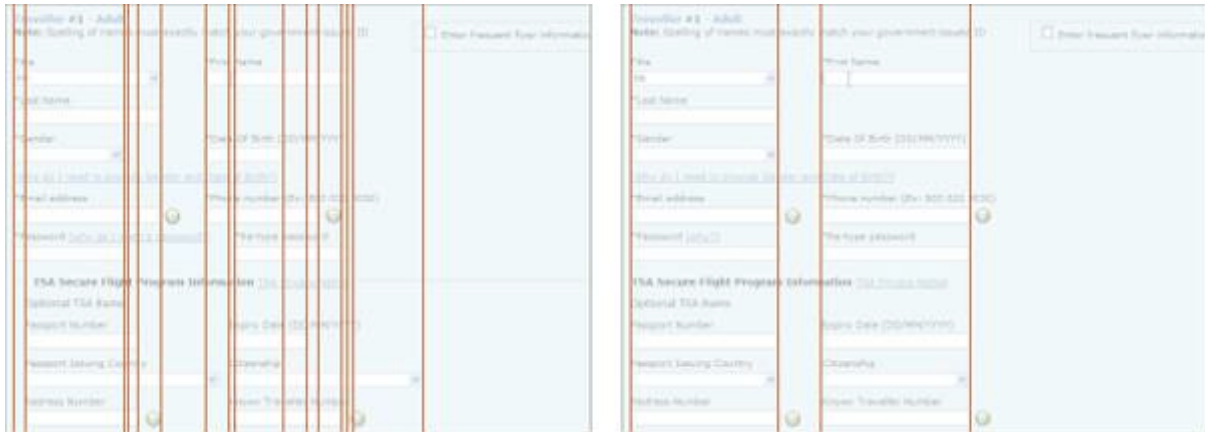
The image displays two side-by-side screenshots of a web form titled "TSA Secure Flight Registration Information". The form contains several input fields such as "First Name", "Last Name", "Gender", "Date of Birth", "Home Address", "Phone Number", "Passport Info", "Report Number", "Report Issuing Country", and "Address Number". In the left screenshot, numerous vertical red lines are drawn across the form, highlighting the misalignment of these fields. In the right screenshot, the same form is shown but with significantly fewer vertical red lines, indicating that the fields are better aligned.

Figure 5: This shows the same forms as in Figure 4 but now the red lines show the vertical alignment points of the various user interface elements. The form on the left (which most people describe as looking more complicated) also has more lines of vertical alignment.

As well as making interfaces look more complicated, poor alignment can also lead to usability mistakes. One of the most dramatic examples comes from a voting paper used in Florida in the 2000 US Presidential election. This ballot paper is called a 'butterfly' ballot because the paper has names down both sides, with a single column of punch holes in the centre. Alignment issues caused many people to vote for the wrong candidate.

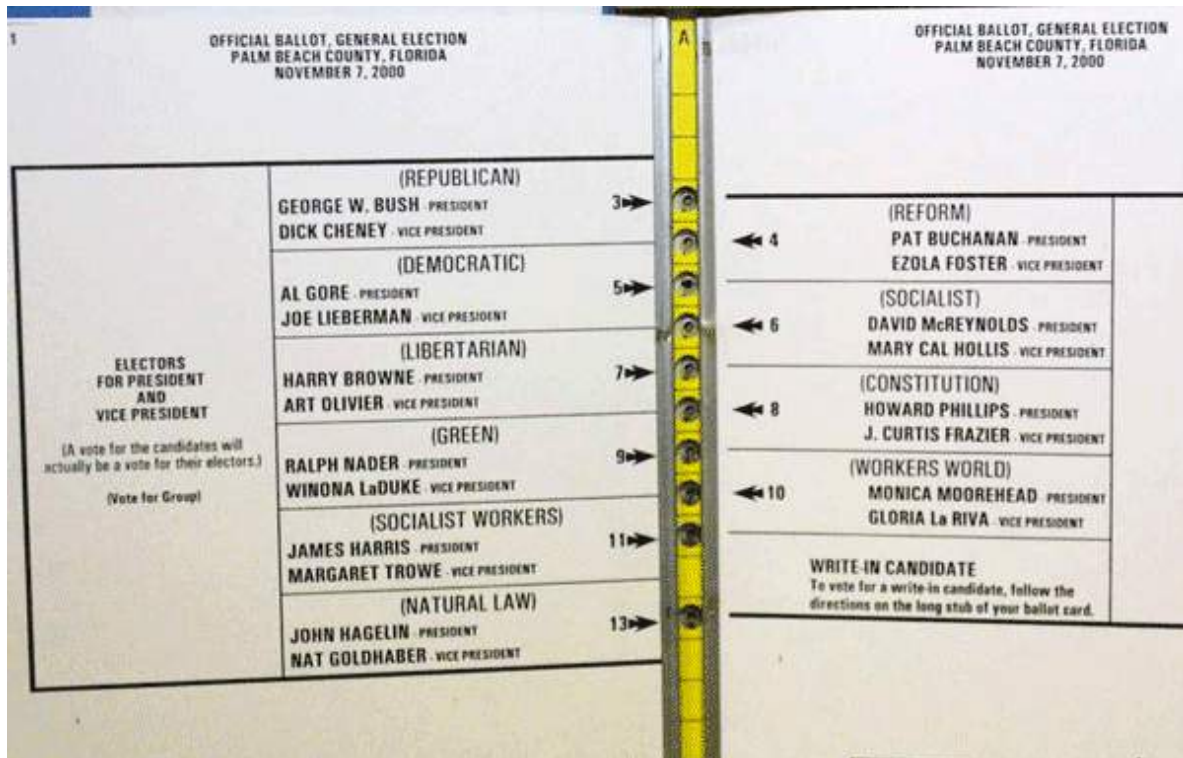


Figure 6: The ballot paper from the US Presidential election in Florida in 2000. Although the Democrats are listed second in the column on the left, voters need to select the third hole on the ballot to vote for them. Voters that select the second hole cast their vote for the Reform party. From Wikipedia.

Proximity

The principle of proximity was first articulated by the Gestalt school of psychology. The Gestalt theorists emphasised visual perception was about perceiving organised wholes, not just about seeing isolated objects. The principle of proximity means that if you place elements in a user interface near each other, people will think that they are related somehow. There are two ways that you can use this principle to enhance usability. First, proximity will help users find the option they are looking for. If users want to see your range of laptops, it will be quicker for them to find this if it's grouped with other related items in a part of the interface devoted to "Computers & Office" than if

it's & one of many options in a disorganised mess on the page (see Figure 7).

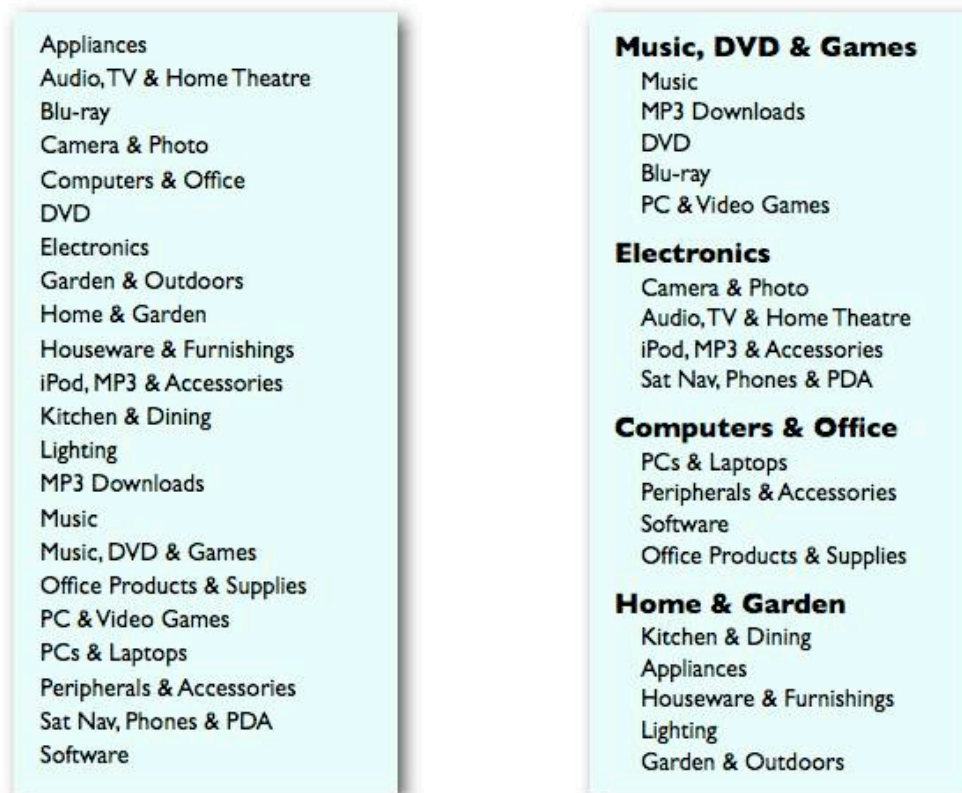


Figure 7: A simple example showing the principle of proximity used to organise items in a navigation menu. The items on the left are in alphabetical order, whereas the ones on the right are organised into groups with each group given a heading.

You can apply this principle in a user interface by appropriate use of background colours, borders and white space: these will help users identify a set of items as a discrete functional block.

But there's a second reason why this principle makes user interfaces more usable: the way you organise information on the page helps people build a conceptual model of how the interface is structured. The web design trend of using

“portlets” (as on the BBC’s home page) is a good example of this.

CRAP designs are more usable designs

It’s tempting to dismiss visual design as mere icing or eye candy. And certainly, you will never rescue an awful interface simply by making it more attractive. (“You can put lipstick on a pig,” as the old saying goes, “but it’s still a pig”). But its wrong to think of good visual design as doing nothing more than encouraging first use. Following CRAP design principles actually makes systems more usable.

7 myths about paper prototyping

David Travis

Paper prototyping is probably the best tool we have to design great user experiences. It allows you to involve users early in the design process, shows you how people will use your system before you've written any code, and supports iterative design. So why are some design teams still resistant to using it? Here are 7 objections I've heard to paper prototyping and why each one is mistaken.

Myth 1: "I can't draw well enough to create a paper prototype."

When given a pencil and paper, most people (myself included) will tend to doodle words rather than draw pictures. Art was never my subject at school, so when I see wonderful, artistic renditions of user interfaces, I tend to feel somewhat humble. I can't draw like that, my inner voice reminds me, so don't pretend you can sketch a user interface.

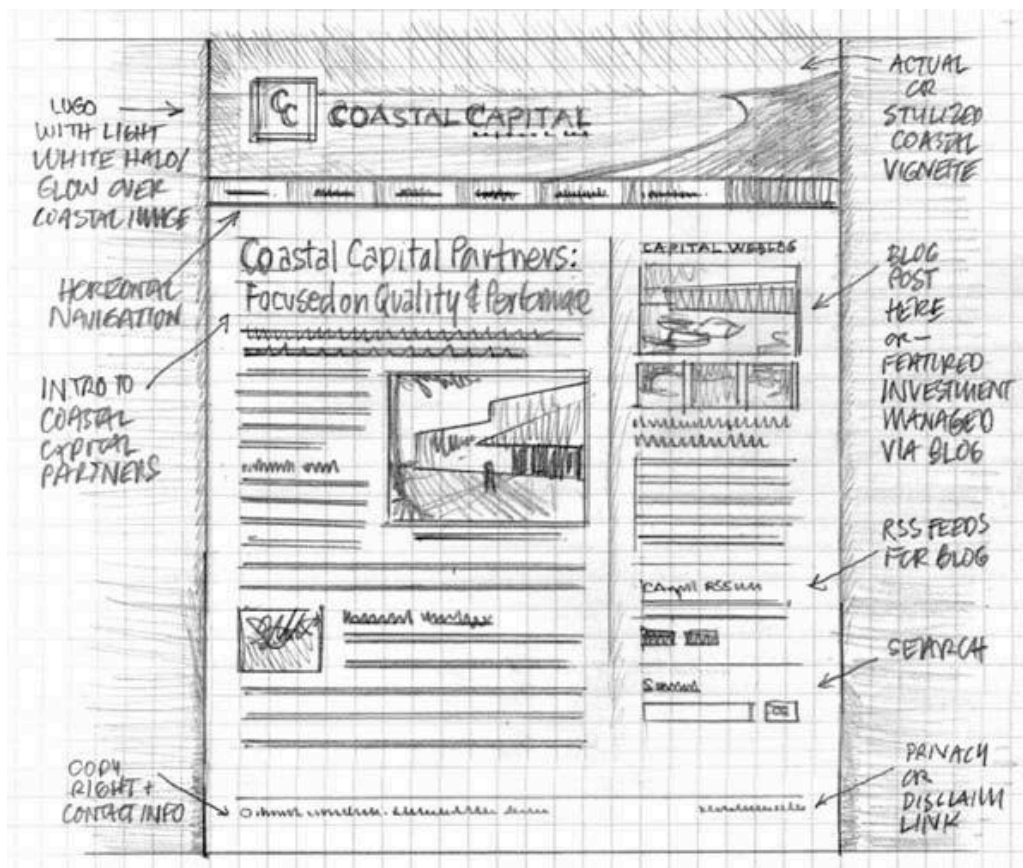


Figure 1: A beautiful prototype on paper, but not a paper prototype. By [Mike Rhode on flickr](#) (some rights reserved)

But just because its on paper, it doesn't mean it's a paper prototype.

A paper prototype is a sketch — a quick visualisation of your idea. The artistic merit of your sketch is irrelevant. The purpose of paper prototyping is to evaluate the idea behind the user interface, not the sketch itself. Paper prototypes are quite literally 'artless': simple and unaffected.



Figure 2: A paper prototype. It's not pretty but the whole prototype was created in minutes by one of the delegates on our Web usability training course.

This focus on 'neat and pretty' is one of the biggest obstacles to overcome with paper prototyping. It has even spawned a real industry. For example, search the web and you'll find imaginative stencil sets for iPhone, web, iPad and Android. These kind of tools simply feed the paranoia of people who worry their sketches aren't pretty enough.



Figure 3: User interface stencils like these give confidence to the artistically challenged but reinforce the myth that paper prototypes need to be precise and accurately drawn. By [Luca Mascaro on flickr](#) (some rights reserved)

Myth 2: “Wireframes are the same as paper prototypes.”

Wireframes show a skeleton view of a user interface. They identify the areas of the screen where content will appear, such as images, text and navigation. Wireframes often include call-outs and notes explaining certain elements of the design in more detail. They are a useful communication tool for design agencies who want their clients to sign off on a design without getting hung up on the colours and images displayed in the user interface.

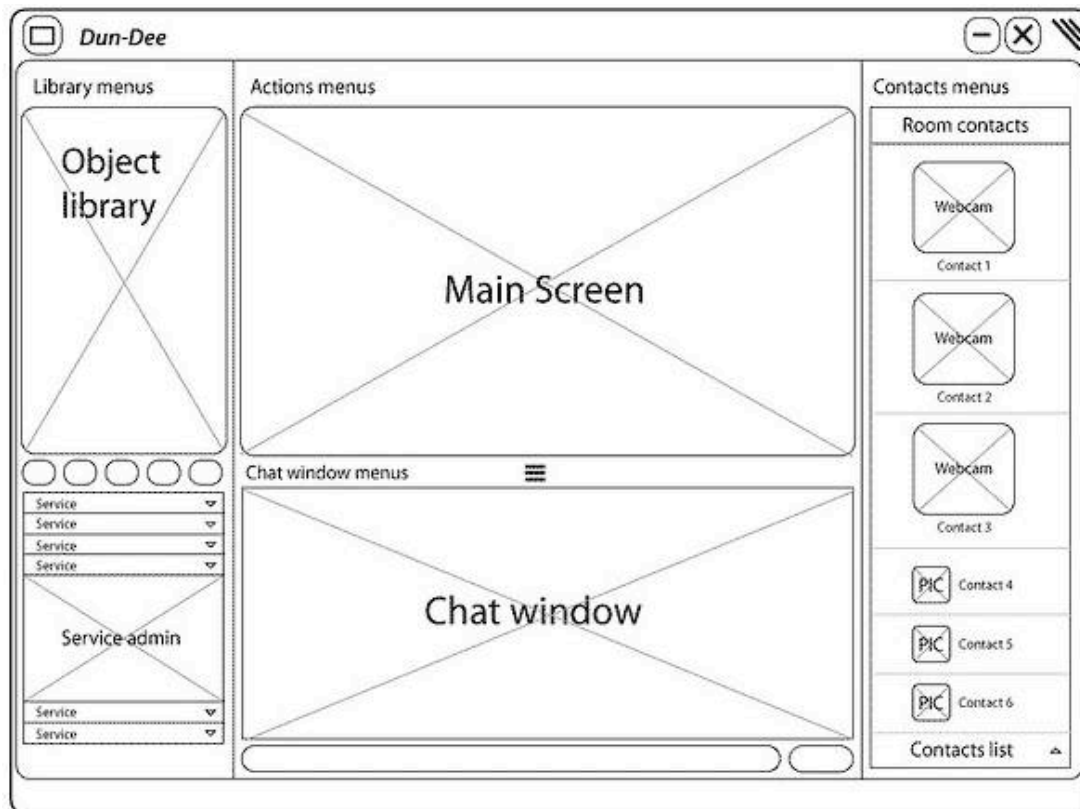


Figure 4: Some designers find wireframes a useful way to communicate a design to their clients. But wireframes are not paper prototypes. By [Sebastian Heit on flickr](#) (some rights reserved)

But wireframes don't work as paper prototypes for at least two reasons:

- They encourage too much focus on the layout of pages. As a consequence they can attract nitpicky feedback on issues like alignment and fonts. In the early design phase you're more interested in the navigation, workflow, terminology and functionality than in details of the visual design.
- Wireframes are usually produced towards the end of the initial design phase, not at the beginning. At this point, many important design decisions — about functionality, for example — have already been made.

Many information architects are obsessed with wireframes but in most projects, they aren't needed: they are just a bureaucratic overhead. You'll find that an electronic prototype, modified as the project progresses, provides a much more useful design communication tool than a wireframe.

Myth 3: "I can do it just as well with Visio."

I've often heard developers object to paper prototyping because they can mock up an interface in Visio, or another favourite tool, in the same time it takes to sketch one out. Developers point out that electronic prototyping scores over paper because it is easier to cut and paste repeating elements (like navigation bars) rather than go through the exercise of re-sketching them. And electronic prototypes can be shown to users over the Internet and tested remotely.

Ironically, 'cut and paste' is exactly what paper is good for — you just need real scissors and glue. You can quickly duplicate elements of a paper prototype with a photocopier or even mock-up the repeating elements in Visio and glue them to your sketched user interface. (If you'd like a starter kit, you can download and print a [paper prototyping helper kit from the Userfocus web site](#). This provides more flexibility than a stencil yet doesn't express the same imperative to draw neatly).

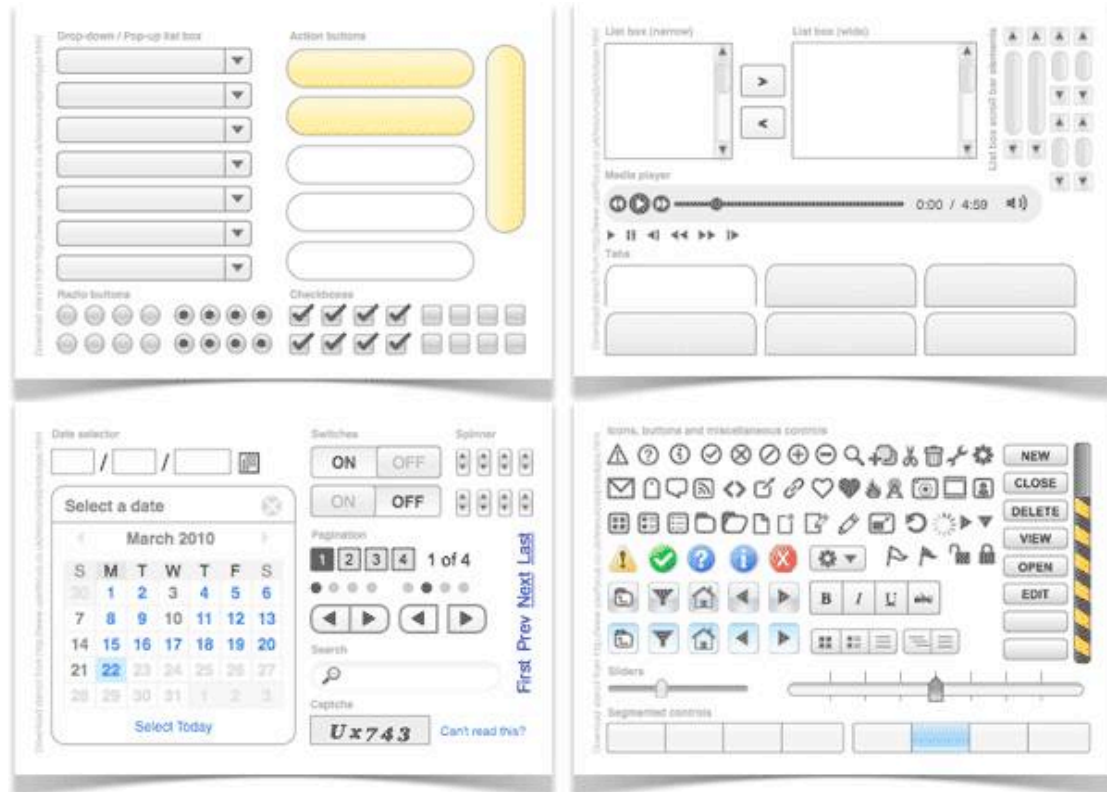


Figure 5: Mock up repeating elements in your preferred drawing application (these were done in OmniGraffle), cut them out and glue them onto your paper prototype. From the Userfocus paper prototyping helper kit.

Similarly, paper prototypes can be tested remotely: you simply scan in your sketches and chain the screens together in PowerPoint or Keynote. You can even record the sessions with usability testing software like Morae.

There is certainly a time for prototyping with electronic tools — but that time isn't the early design phase, for a couple of reasons.

- Electronic tools constrain your thinking about what can be achieved. So if you don't have a 'carousel' widget you'll find you don't include that interaction idea in your electronic prototype.
- Electronic prototyping blinkers you into thinking only about incremental improvement. In the early design

phase you want to generate a number of completely different ideas, test them, and combine the best parts into a converged solution. Bill Buxton makes the point that you first need to get the right design before you move onto getting the design right. Electronic tools aren't good at supporting the divergent thinking you need to get the right design.

I think there's a deeper issue here, to do with confidence. Developers feel happy in front of a computer: it's their domain and it's what they're skilled at. Take away their computer and they feel a bit... naked. But this massively undervalues another key skill that developers have: problem solving. As the development team at Red Gate Software point out, more developers should realise that their skills extend way beyond that of the keyboard.

Myth 4: "Whiteboarding is just as effective."

Many designers I've spoken with tell me that they already do paper prototyping — they just do it on a white board or flip chart instead. The design team then discuss the ideas and refine them before moving to implementation.

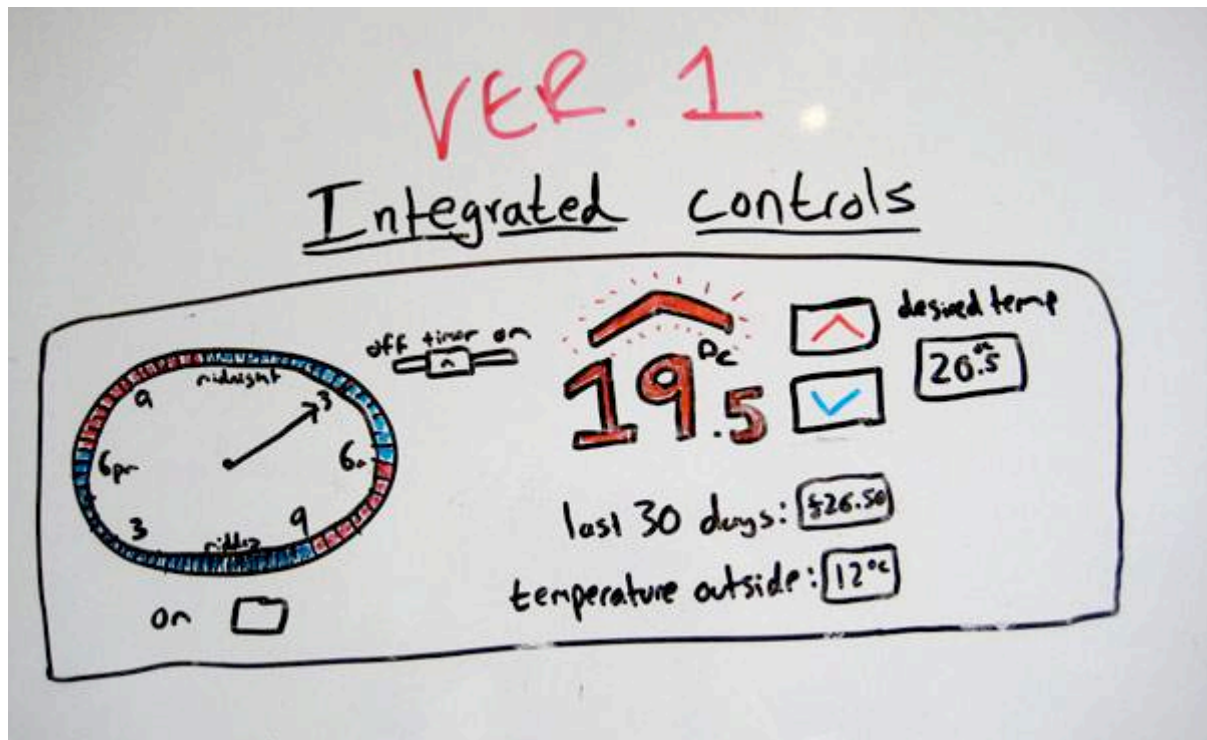


Figure 6: Using a whiteboard to generate design ideas like this one is a good way to brainstorm but not a good way to prototype, because you can't test out the interface. By [Frankie Roberto on flickr](#) (some rights reserved).

Whiteboarding, and its close cousin sketchboarding, are terrific creativity tools, used by design heavyweights like Adaptive Path — but they're not paper prototyping. These techniques remind me of Gerry McGovern's joke that the worst possible way to design a web site is to have five smart people in a room drinking lattes and discussing branding. The purpose of paper prototyping is to quickly generate design ideas that you can test out with users. It's not for coming up with design ideas that can be voted on by the design team. This is because paper prototyping isn't just about design — it's about iterative design: creating stuff you can test quickly and then discard or improve upon.



Figure 7: Sketchboarding is a wonderful creativity tool, but it's not paper prototyping because the designs aren't tested with users. By [Mack Male on flickr](#) (some rights reserved).

Rather than hold a fruitless design meeting where you spend hours arguing over where you should put the navigation, you could spend the time more productively by creating a couple of different paper prototypes and testing out the alternatives with a handful of users. Not only will this help you make a decision quickly, you'll also have the confidence that the final decision is based on real data rather than on a designer's intuition, or worse: the HIPPO (the highest-paid person's personal opinion).

Myth 5: "Users behave differently with a paper prototype than with a real system."

Paper prototypes use the participant's finger as the mouse and a pen as a keyboard. Interfaces are roughly drawn and are

clearly not finished prototypes. This is clearly different to a real system and so a common objection is that users will behave differently with a paper prototype.

This objection has no basis in fact. A raft of published research shows that paper prototypes are just as effective as high-fidelity prototypes at detecting many types of usability issues.

Myth 6: “It looks unprofessional.”

There is no denying that paper prototypes lack beauty — but this is different from saying that paper prototyping is unprofessional. In my experience, so long as you get participants really using the paper prototype (rather than sitting back and having it demonstrated to them), they quickly get ‘lost’ in the experience of using a paper prototype. This makes issues of professionalism moot.

In contrast, I would argue that your users will think you’re more professional because you’re clearly involving them early in the design process and taking their comments seriously.

If your management is still unsure, run a trial with single participant and ask, “Is your opinion of our organisation better or worse after this experience?” I’ll bet you’ll hear only positives.

Myth 7: “I can’t prototype interactivity.”

It’s true that there are some intricate interaction styles that don’t lend themselves to paper prototyping. Gesture-based interactions on the iPad and iPhone can’t be replicated on paper. Panning and zooming, Google Maps-style, isn’t practical. Similarly, applications that require continuous

scrolling or that include long download or response times can't be properly emulated.

But the vast majority of web and software interaction — even on an iPhone — is still point and click (with a little keyboard too). These can easily be emulated with a paper prototype. Rather than focus on the edge case that won't work, think of the vast majority of situations where paper prototyping will work.

Help! What The Beatles can teach us about writing support material

Philip Hodgson

Reading user instructions continues to rank high on people's lists of 'activities-to-be-avoided-at-all-possible-costs'. We've worked with a number of clients to improve their user support materials and we frequently encounter five common mistakes made by development teams. This work has given us some insight into how best to avoid these problems occurring in the first place.

'Help!' cried John, Paul, George and Ringo, in a moment of despair (even going so far as to spell it out in semaphore on an album cover). Yes, they needed somebody, and not just anybody. When they were younger (so much younger than today) they never needed anybody's help in any way. But now those days are gone, and we live in a world where complicated products are accompanied by spectacularly awful user instructions — so now they are not feeling so self-assured.

I know the feeling. My heart always sinks if I have to read a user manual. I know I'm in for a bumpy ride.

What's your experience with user instructions? If you're like most people, you read instructions only as a last resort, or maybe never. "We just want to take things out of the box, turn them on and see them leap into action without having to read anything," admits Rory Cellan-Jones, BBC's technology correspondent. He's not alone — in a survey of 75,000

technology users, conducted by Gadget Helpline, 64% of men and 24% of women admitted that they did not read instructions before phoning support.

As it turns out, it's not true that people don't want to read instructions. As Kathy Sierra has pointed out, they just don't want to read yours. Paradoxically, users seem quite happy to turn for help to third-party technical support manuals. From 'Guides for Dummies' to 'Missing Manuals', there are shelf-loads of them in every bookstore. And increasingly people are turning to user-generated content, blogs and online discussions for support with the latest application or device. 'RTFM' has been usurped by 'just Google it'.

But why do user manuals and other support materials have such a bad reputation, and why are they often so useless? I've written previously about how to write better instructions but the real problems go deeper than issues of technical writing. And these problems are amazingly consistent. For example, I've sat through hundreds of hours of usability tests of help systems and carried out dozens of expert reviews of support materials on systems as wide ranging as medical devices, banking software, mobile phones, e-commerce sites, intranets and even large manufacturing systems. A handful of common mistakes occur time and time again.

Here are 5 guidelines for avoiding these mistakes:

1. Solve the right problem
2. Find out where your users struggle
3. Know the skill level of your users
4. Focus on red routes
5. Don't isolate the technical writers

In this article we'll take a look at each of these mistakes and suggest ways to avoid making them.

But first a note about terminology: user instructions and help systems come in a range of formats. So in addition to the traditional user or operator manuals (aka user guides) or instruction sheets, we can include installation guides, video instructions, interactive demonstrations, online instructions, built-in help systems, quick-start reference cards, pop-up user tips, trouble-shooting guides, FAQs, instructions printed on packaging, training materials, and support-center crib sheets. Because the format is not the issue here, in this article I'm going to take a lead from the Fab Four and refer to any and all of the aforementioned simply as Help!

1 Solve the right problem

At the risk of stating the obvious, the main reason for creating 'Help!' is to help the end-user to get things done, to help them learn the system, to help them get past tricky obstacles and to help them build confidence.

It's about helping the user. The clue's in the name.

Rather like a safety net, a lifeboat, or an insurance policy, people would prefer not to use 'Help!' but when it's needed it must do its job effectively and efficiently, be accessible and easy to understand. The last thing anyone wants when grappling with a frustrating interface is a set of instructions that makes things worse, or instructions that need other instructions to explain them. I once usability tested a system that had eleven 'Help!' resources, some of them propping up the application and some of them propping up each other.

In spite of this self-evident fact, most ‘Help!’ projects I’ve consulted on turned out not to be focused on the goal of improving the users’ experience. The clients who commissioned me said this was the goal but when we got down to brass tacks the real problem the teams were trying to solve was how to reduce the costs of the support materials. The real problems were more like: ‘How can we redesign the materials without using color?’ ‘How can we put these instructions online?’ And most frequently: ‘How can we reduce the page count?’

Similarly, I’ve also encountered ‘Help!’ that is really a litany of obligatory cautions and warnings, or a place to dump stuff previously overlooked — a last minute attempt to write the product out of trouble.

Things you can do to avoid making this mistake

- Put the user first. Adopt a user-centred design process such as the one described in ISO 9241-210.
- Create a user experience vision for ‘Help!’ just like you have a user experience vision for the system.
- Assign high priority to ‘Help!’ from the outset. Consider it a critical part of your system, not an afterthought.
- Start working on ‘Help!’ early and develop it in lockstep with the system.

2 Find out where your users struggle

When I’m reviewing a client’s ‘Help!’ system, I often find that the product or system itself has never undergone any kind of usability testing. This sets off alarm bells.

How can you give your users help if you don't know where they struggle?

You can't, of course. That's why the default approach is to instruct the user on everything — every step of every function and every feature — and to assign everything equal weight. It is as though the writers are telling the user: "Look, we've designed a really complicated product here. Frankly, you're not going to be able to figure it out on your own so you're going to need all the help you can get."

This often happens when the writing team doesn't know the system interface well enough to know what to focus on. So they play safe and cover everything.

This 'throw everything at them' approach is akin to a cavalry charge: really impressive to look at but disastrously ineffective at accomplishing anything. It typically results in an overwhelming 'Help!' system that is difficult to navigate, with the really important things lost in the noise. The net outcome tends to be 'Help!' that ironically makes things worse and creates a negative first impression. Needless to say, it also increases development time and costs.

'Help!' should be commensurate with the ease of use of the user interface. A well designed user interface that has had most of its usability problems fixed will require only minimal 'Help!' — witness the Apple iPhone which comes with no 'Help!' at all. It also means that you can make a much more encouraging first impression: "This is a such an easy product to use we've given you a simple user guide just to cover one or two potentially tricky actions".

Things you can do to avoid making this mistake

- Test the usability of the system and invite the writing team to attend.
- Prioritize the usability problems
- Structure ‘Help!’ around the most severe usability problems — there’s no need to write instructions for things that are obvious.
- Test the usability of ‘Help!’ This requires test participants to explicitly follow the ‘Help!’ instructions. Written instructions are easy to change on the fly so you can iterate quickly during a single test.

3 Know the skill level of your users

Another reason why many writing teams are hesitant to leave anything out is because they don’t know the typical user’s skill level. Many document writers have never met a real end user, and have never seen people using the product or application for which they’re writing support.

This is rarely the fault of the technical writers — often the product design team has never met a real user either. Many development teams know the user only through high level demographic data handed down by marketing. But knowing someone’s age, gender, household income, and number of kids tells you nothing about their comfort level with technology, their domain knowledge or their experience with similar products. Are your readers going to be total novices, intermediates or experts?

Knowing your audience is mission critical for a writer. But a default approach I often see can be characterized as: ‘Play safe.

Assume everyone using the product will be clueless. Cover all the bases. More is better than less.’ As a result, you’ll often see ‘Help!’ that gives step-by-step instructions on how to plug a battery charger into a wall socket, or describes every step in a sequence required to access a menu of options — only to describe exactly the same steps all over again for every option in the menu.

Even more frustrating is ‘Help!’ with a split personality. It explains easy operations in excessive detail but then glosses over advanced operations, as though the user has suddenly morphed from a dunce to an MIT computer scientist. This misses the mark in both cases. When I’ve come across this on client projects I sometimes discover that the reason the advanced operations were poorly described is because the writers themselves didn’t understand these functions — and made up for the lack of content by expounding at great length on the more basic features. Other times, it’s often a sign that ‘Help!’ was written before the complex elements were actually finished.

Understanding the skill level of your user is critical because it helps you prioritize instructions. It tells you what to spend more time on or what to explain using a different technique. It also gives you the confidence to know what to leave out. For example, anyone who uses a mobile phone every day does not need half-a-page of painstaking instructions explaining how to adjust the volume on an application.

Things you can do to avoid making this mistake

- Find out about your users. Don’t be satisfied with just demographics, identify their skill level and focus on how they do their work.

- Create user personas. Use them as your target audience. (Warning: don't just make up user personas, base them on real data).
- Carry out a card sort of the 'Help!' elements — this will improve the structure of the materials and teach you the users' terminology so you can make sure you are speaking their language.
- Credit the user with some smarts. And if you're writing 'Help!' for a system that requires a high level of expertise to operate — credit them with lots of smarts.

4 Focus on red routes

Most 'Help!' tends to describe the product's features. This is a mistake. Typically, the user is not thinking about your product or system at the level of features: they are trying to accomplish a task, and tasks usually consist of actions strung together like beads on a string. Think why someone has turned to support and you quickly realize that 'Help!' should focus on helping users achieve their goals. This means structuring 'Help!' around the main user journeys or red routes.

When you don't structure 'Help!' around red routes, functions and features tend to get presented piecemeal and out of sync. Users then need to pogo-stick around the 'Help!' materials to solve their problem.

Just as it's important for the design team to know the red routes, it's essential for the technical writers to know them too so that 'Help!' instructions can form a coherent end-to-end flow through the task.

Things you can do to avoid making this mistake

- Identify the red routes and make sure the writing team knows them too.
- Work with the UX team to learn which tasks are most important to the users and which are most frequently carried out. Make sure 'Help!' covers these tasks.
- Learn the users' mental model (their expectations) for the main tasks.
- Carry out a Cognitive Walkthrough for each of the main tasks.

5 Don't isolate the technical writers

In over 20 years of consulting, I've only ever once attended a product development meeting with a technical writer present. I've had meetings with technical writers separately from design teams, usually in some other building, or even in another city, and often just prior to product launch, but only once in an actual core team meeting.

One reason for this, of course, is that many companies, especially smaller ones, don't have dedicated technical writers and have chosen not to employ people who are experts in creating effective user documentation, instead assigning the job to a software programmer, an engineer, a designer — or even to an intern.

There are lots of reasons why it is a good idea to hire expert writers, but the most important one is that the people creating 'Help!' must be able to see the system through the eyes of the end user. It's true that a software programmer or an engineer

may well understand the system far better than a technical writer — but that's often a handicap, not an asset.

It gets worse. I've also met technical writers who have never actually used the product they are writing about — even some who haven't even seen the product. These poor souls have to work by changing a few instructions in documents from an earlier release. I think this is a bit like a blind date where you only ever communicate with your partner by text message. This scenario defies logic and common sense.

First and foremost, technical writers are problem solvers, and they should be an integral part of the core design team from the outset. If you don't have a technical writer you need to hire one and put the product in his or her hands. Introduce your technical writers to your user experience and user interface designers so that they can work closely together. And insist that they attend design meetings. The person who writes your 'Help!' needs to be part of the design team's thinking and decision making, not someone you bring in late in the game to rehash old user guides.

Things you can do to avoid making this mistake

- Make technical writers part of the core design team.
- Give the writers the system to use so they can experience its strengths and weaknesses for themselves.
- Invite the writers to attend usability tests of the system and usability tests of the 'Help!' materials.
- Introduce the writers to your user experience specialists and to your user interface designers.

Create useful 'Help!' not useless 'NUJV'

So... What can The Beatles teach us about writing support material?

It turns out The Beatles weren't really signalling 'Help!' on their eponymous album cover. I was lying. What they were signalling was that well known phrase, 'NUJV'. It turns out that the semaphore arm positions for 'Help!' didn't look aesthetically pleasing to the photographer so he rearranged them to form a nonsense phrase.

Don't make the same mistake. Create useful 'Help!', not useless nonsense.

Confessions of an Axure Master: 5 shortcuts for laying out Axure pages in record time

Ritch Macefield

One of the quickest ways to speed up your use of software is to learn shortcuts. Here are 5 little-used shortcuts for Axure that will help you work like an expert.

Amanda, the new Axure apprentice, stood silent as FuYoda teed off on the short par 3 at the new “Design Space” Golf course. As FuYoda’s ball began rolling slowly yet surely across the green for yet another hole in one, the apprentice started a conversation.

“Great shot FuYoda,” said Amanda. “Your golf skills are like your Axure skills: three times faster than most people’s”.

The Master replied: “Hummm, more quick with Axure, more time for golf like this”.

“I think it will take me a long time to learn to do things so effortlessly,” said the apprentice.

“Hummm,” said the Master, “Some things take time —some things shortcuts have. Like page layout.”

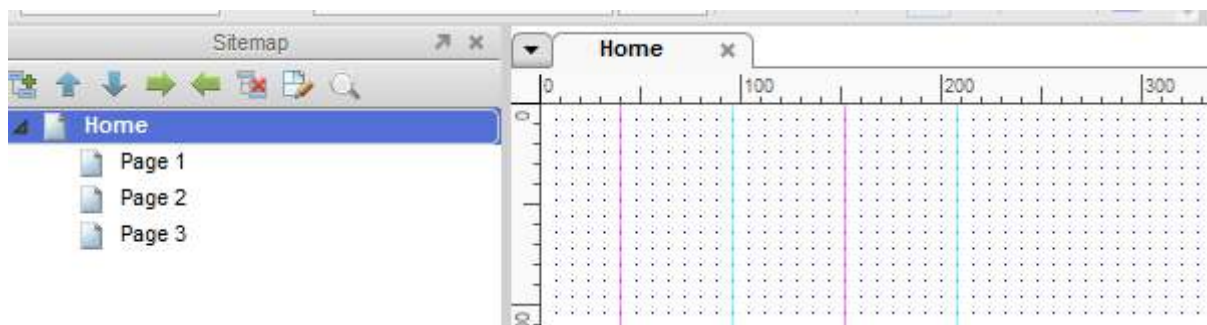
“Please Master,” said the apprentice, “Tell me your page layout tips for Axure.”

OK, enough with the Yoda language. Let's look at 5 shortcuts for laying out Axure pages in record time.

Two types of guide

When lining widgets up on pages, most Axure users know that you save a lot of time by dragging on a guide from page rulers with the mouse. Not only do guides provide a visual cue but, by default, Axure widgets snap to these guides as well. By default, these guides are blue and they appear only on the page where they were created.

However, if you hold down the Ctrl key during the drag you end up with a red guide. These guides appear on every page, including masters and all dynamic panel states. This makes them ideal for lining up widgets across multiple pages. Global guides can also be imported from other Axure files if you have spent a long time setting them up or you want to use them to help implement a style guide or standard page formats.

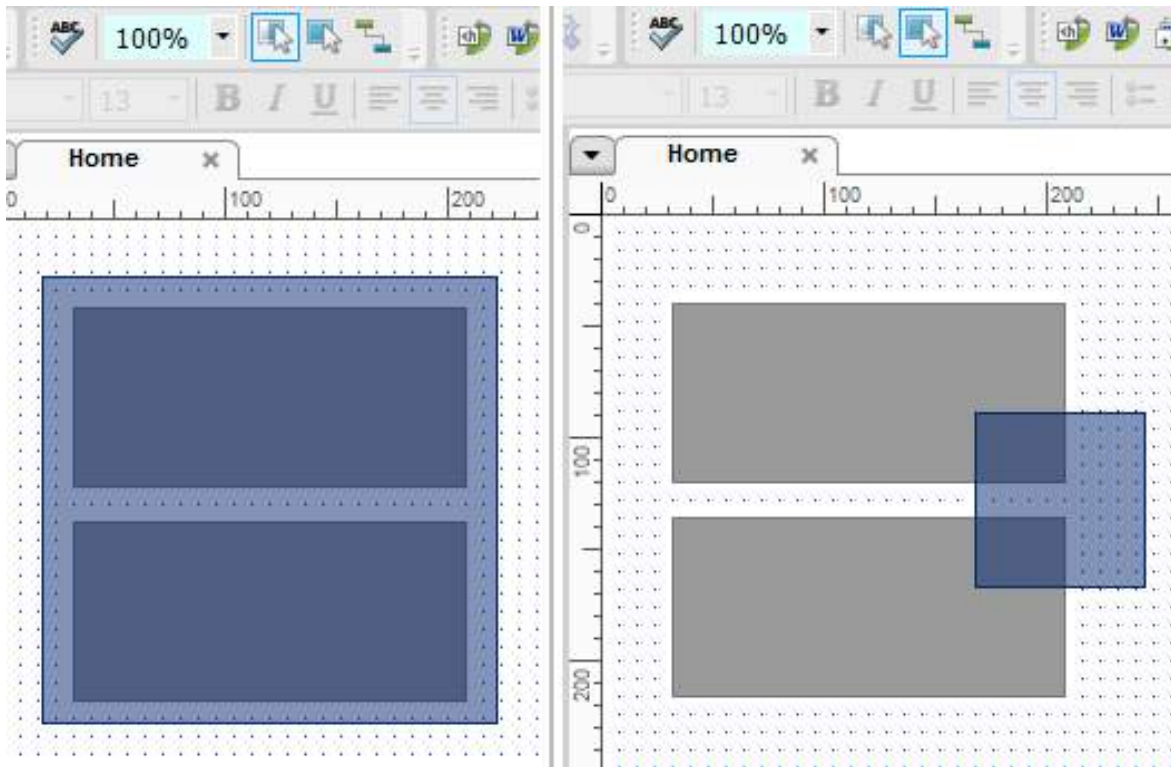


Axure's two types of guide.

Two types of selection mode

Axure has two selection modes: 'Contained' and 'Intersected'. In 'contained' mode, when you draw a selection box, every widget totally inside the selection box is selected. In

‘intersected’ mode, any widget that is touched by the selection box is selected.



Axure's two types of selection mode: 'Contained' (left) and 'Intersected' (right).

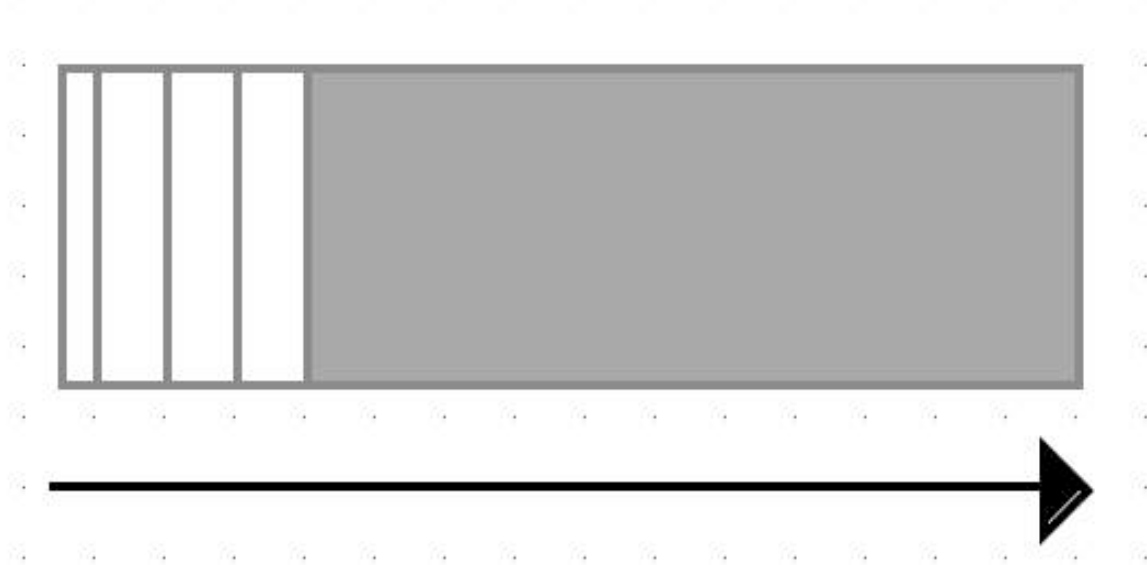
When pages get complex, alternating between these modes makes selecting multiple widgets much faster. In fact, sometimes it's pretty much the only way to make the correct selection. You can quickly alternate between these modes using the keyboard shortcuts: F9 for intersected and Ctrl-F9 for contained mode.

Use Shift-Arrow

Most Axure users know that you can 'nudge' a widget (or set of selected widgets) one pixel at a time using the arrow keys on the keyboard.

But if you hold down the Shift key at the same time, Axure will first nudge the widget so that it lines up on the grid in the

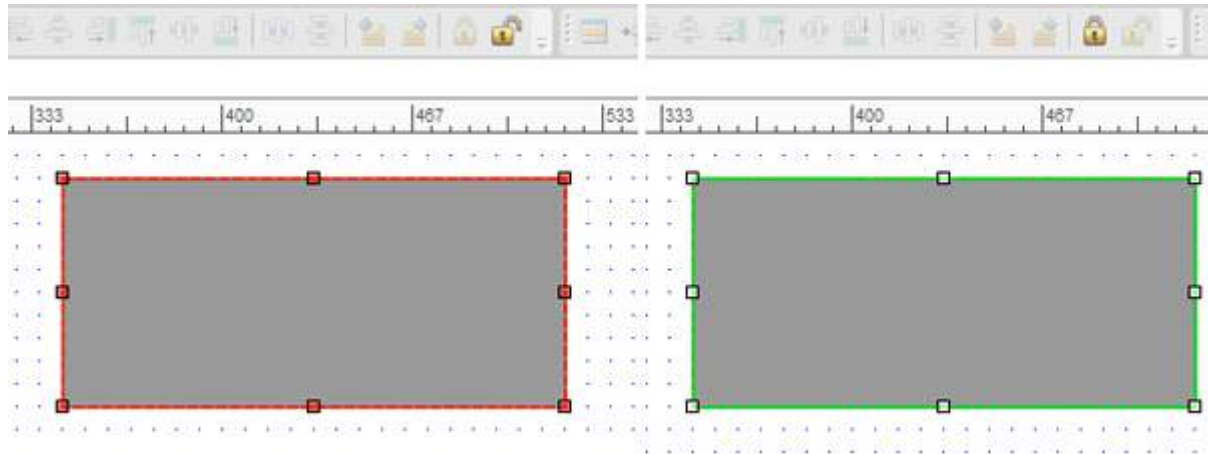
direction of the nudge (if its not already lined up in that direction). Subsequent nudges will move the widget one whole grid space at a time.



If you 'nudge' with the shift key held down, widgets line up with the grid.

Lock widgets

If you have spent time carefully positioning a widget you can lock it using the padlock icon (or Ctrl-K). You can also lock a whole selection of widgets. Locked widgets have a red border and control handles and cannot be moved with the mouse. Use the unlock icon (or Shift-Ctrl-K) to unlock widgets.

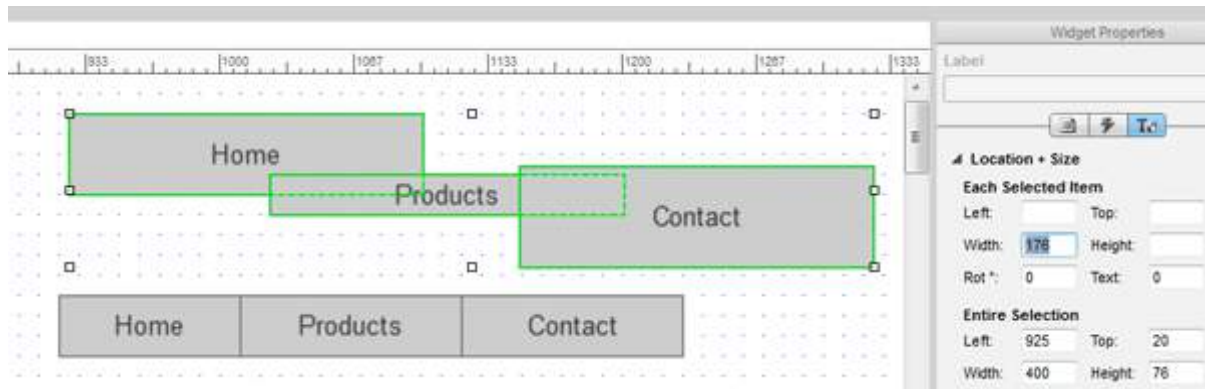


Widgets can be locked (left) or unlocked (right).

Use typed input to resize and/or position multiple widgets at the same time

Very often, we want to make several widgets the same (specific) size or maybe just the same height or width. Likewise, we often want to give multiple widgets the same X or Y position on the page. A good example of this is when laying out a top menu bar. We want all the widgets in the menu bar to have the same height and the same Y position (we may or may not want them to all be the same width).

Axure has a simple yet powerful utility you can use to achieve this. Select the widgets you want to resize/position then type in the correct values in the “Location & Size” section in “Formatting” mode within the “Widget Properties” panel.



You can use the “Widget Properties” panel to reposition and resize multiple objects.

Communicating errors

David Travis

Ideally, you'll design your system to prevent errors from occurring in the first place. But no matter how simple your system, someone, somewhere, will make an error when using it. The difference between a great user experience and an awful one is what your system does next.

Interaction design is the art of managing a conversation between a user and a computer system. When a user asks for something that the computer can't understand, the computer throws up an error message. An error message is the computer's way of saying, 'Pardon?'

System errors are critical moments in the user experience, just like complaint handling is a defining moment in the customer experience. Deal with the problem well and your customer stays a customer. Deal with the problem poorly and the customer leaves — and, after he or she starts a FaceBook group to discredit you, you may lose lots of future customers too.

In this article, I'll outline 6 principles for dealing with system errors:

- Be visible
- Be precise

- Give constructive help
- Speak the user's language
- Don't blame the user
- Don't be stupid
- Be visible

Be visible

Once an error has occurred, you need to ensure the user is aware of the problem. This means the error message should be visible: the user shouldn't have to divine if, or where, an error has occurred.

One of my favourite examples of a (near) invisible error message is this dialogue box from an install program. At first glance, everything seems fine and it would be easy to misread the dialog and assume the drivers had been installed:



This dialog box from a install program contains a green tick and a large 'OK' button. The design implies that the drivers have been installed, when they haven't.

A good way to test out this principle with a web site is to submit a form with errors or missing 'required' fields and see what happens next. When users make a mistake with a form, you need to make it absolutely clear where the error has occurred. Returning a form that simply lists all of the problems

doesn't help users appreciate where the error has occurred and what they need to do to fix it.

To resolve the issues thrown up by the web page in the next example, the user needs to continually scroll back to the top of the page to understand which error message refers to which field. A better design would be to place the error message next to each field that needs to be completed.

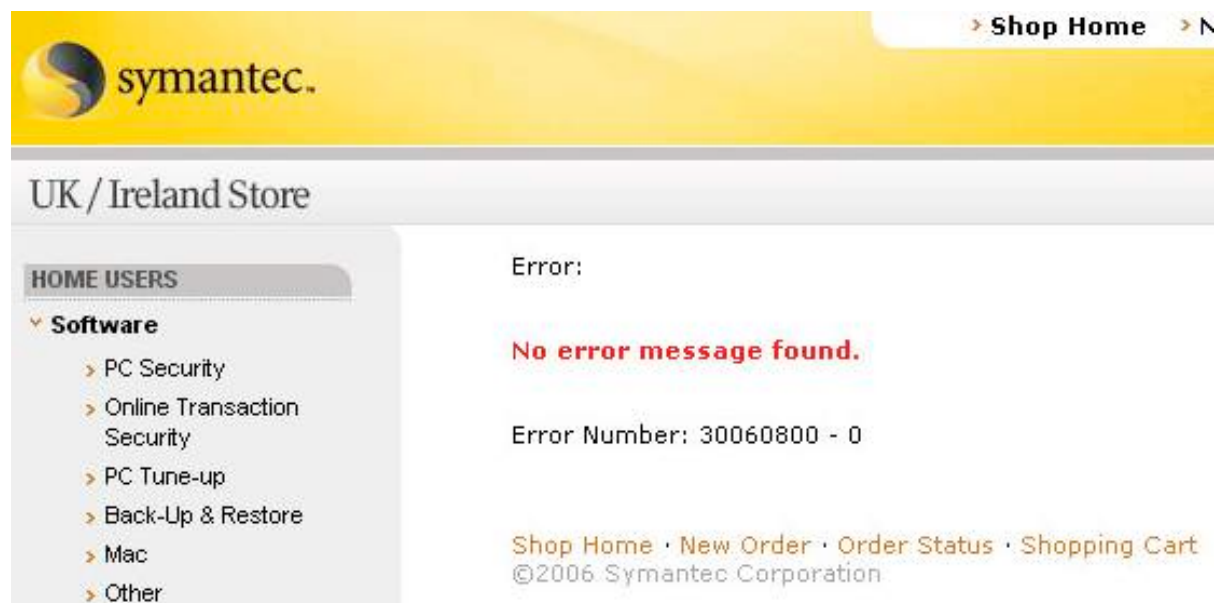
The screenshot shows the Business Link registration page. At the top, there is a navigation bar with the Business Link logo, the text "Practical advice for business", and a "Your account" link with a "Log in" sub-link. Below this is a progress bar with five steps: "About you" (highlighted in green), "Terms & conditions", "Additional details", "Security details", and "Confirmation". The main heading is "Registration". Underneath, a "Problem:" section lists ten error messages in red, such as "You did not choose a title" and "You did not enter your first name". Below the errors is the "About you" section, which includes a note that asterisked information is required. The form fields are: "Title" (a dropdown menu showing "Select title"), "First name", "Last name", "Email address", and "Re-enter your email address". Each field has an asterisk indicating it is required.

This page appears on the Business Link web site when you want to register but don't complete all the fields. A long list of errors appears at the top of the form, and although the error message itself is helpful ('You did not choose a title'), it's not clear which issue relates to which part of the form.

Be precise

A second key principle is that good error messages should be precise and specific. Error messages need to say what has happened, why it's a problem, and how it can be resolved. An error message that simply states, 'an error has occurred' is no use to anyone.

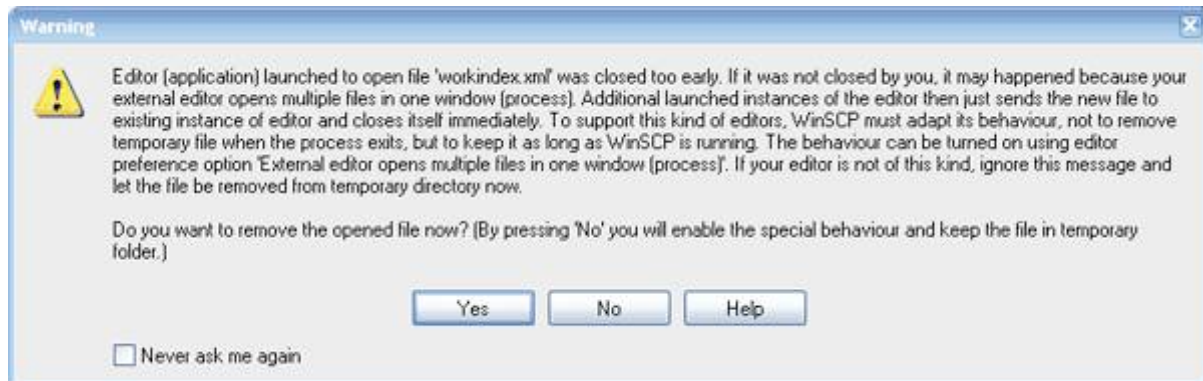
I came across this example when my trial version of Symantec's antivirus software was coming to an end and I wanted to buy a license. I clicked a link in the application and arrived at this page on their web site:



This error page from the Symantec web site contains the text 'Error: No error message found', along with a 9-digit error code. After this, I decided to choose a different supplier for my antivirus software.

But there are limits to how precise and specific you need to be. Remember, it's just an error: the user needs enough information to understand the issue and move on. Don't take this as an opportunity to subject the user to a training course in the use of your program. The example below from WinSCP (a

Windows FTP program) provides too much background on the error:

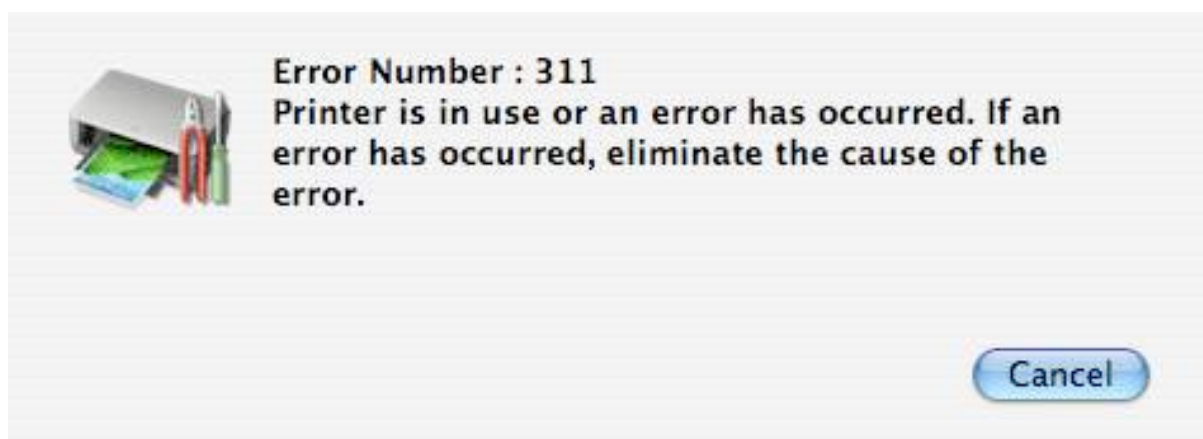


This warning dialog from WinSCP comprises 8 sentences and nearly 150 words. The text itself is also poorly written and ungrammatical. Its only saving grace is the 'Never ask me again' checkbox.

Give constructive help

Nobody likes being stuck. Users encounter errors on the way to achieving goals, so your error message needs to give users the assistance they need to resolve the problem and move on.

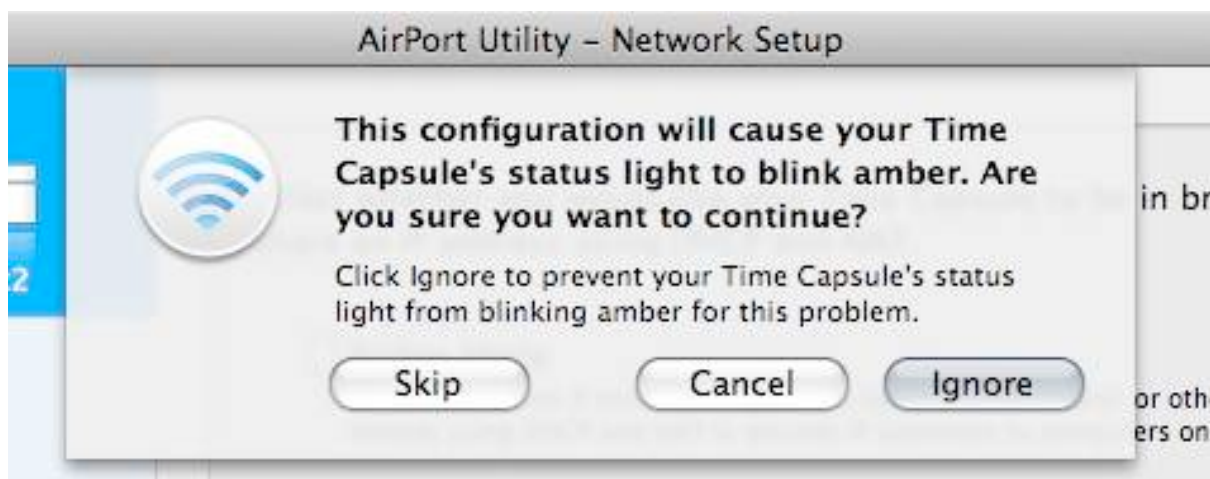
This error message popped up when I was using a Canon printer:



This error message begins with an error number (always a bad sign). The message itself — 'Printer is in use or an error has occurred. If an error has occurred, eliminate the cause of the error' — provides no constructive help.

Simply asking the user to ‘eliminate the cause of the error’ is inadequate. The user needs to know exactly what has gone wrong and what he or she should do to fix it. (And don’t get me started on heading up the message with an error number. Although error numbers can be a useful way for advanced users to troubleshoot a problem, they shouldn’t be the first thing that users are asked to read).

A related issue occurs in an error message that a colleague came across when installing an Apple backup disk on his home network. The consequences of choosing each option is unclear from this dialogue. What is the penalty of ignoring the status light flashing amber? Should the user ‘skip’ or ‘ignore’ this error?



Even Apple struggle with error messages. The text in this message reads: ‘This configuration will cause your Time Capsule’s light to blink amber’. So what? A good error message should provide more constructive help.

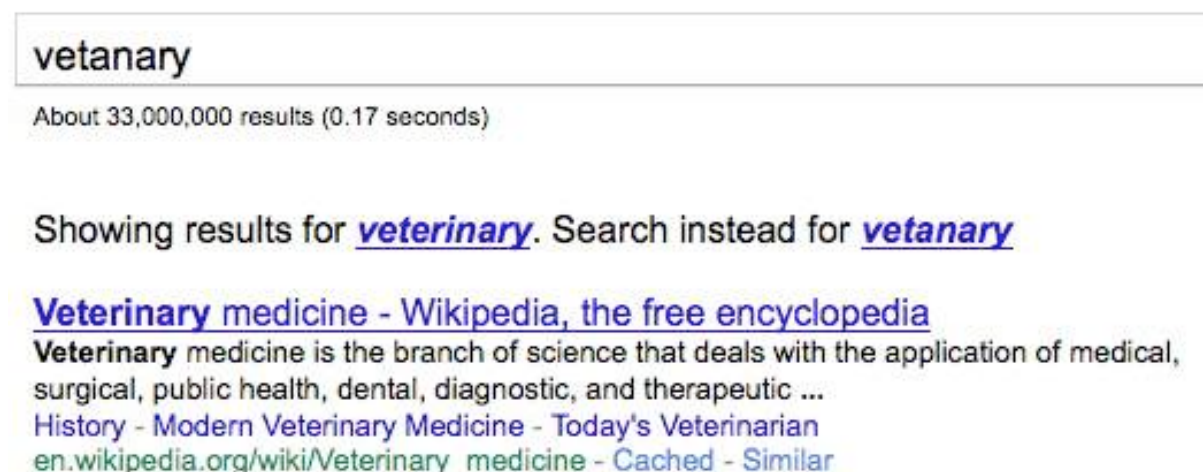
It may not seem like an error message at first, but a ‘no results found’ response to a search enquiry is indeed an error message. The image below shows an example from the BBC web site. In this example, I’ve searched for ‘veterinary’ but spelt it wrongly; the ‘error message’ simply tells me that there

are no matching results. But this isn't very constructive: it's not clear what I should do next:



Searching for 'vetanary' on the BBC web site gives no results but the error message provides no suggestions on what to do next.

A better design would be to provide suggestions on what the user could do next, offer a 'did you mean' suggestion, or even (as shown in this better example from Google) automatically correct the search term:



The same search on Google corrects the spelling.

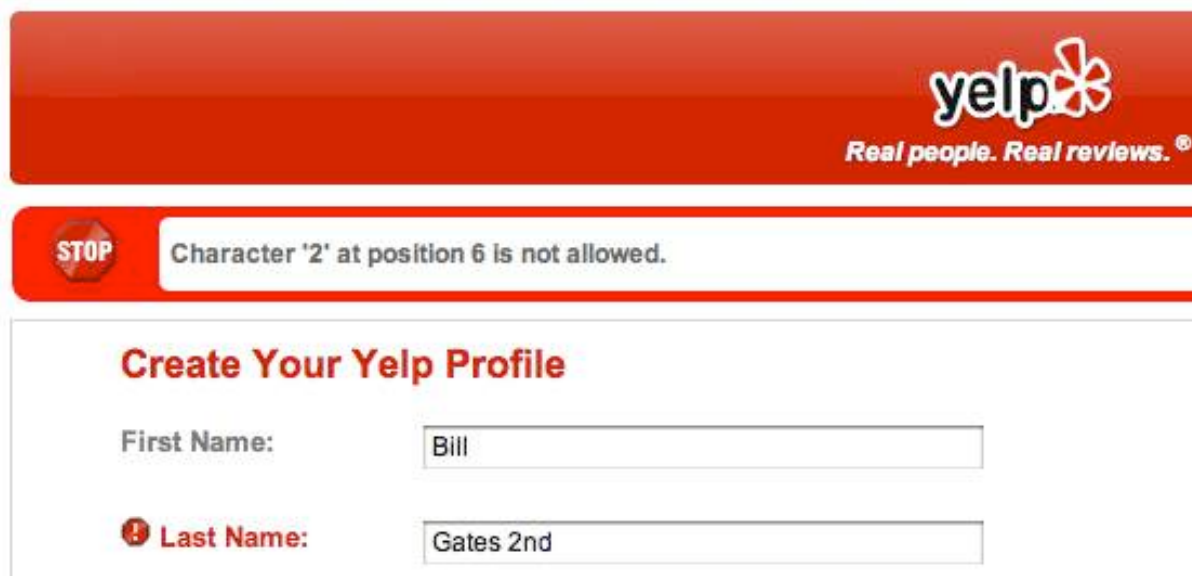
Speak the user's language

I'll admit it: writing good error messages is hardly the most exciting part of system design. Because of this, the precise wording of error messages tends to be left until the last moment and is often left to the person who writes code, rather than the person who writes content. This might be the reason

why they are often so poorly phrased. Error messages often appear written from the system's perspective rather than from the user's perspective.

This example below comes from Yelp. When creating a Yelp profile I've included a number in the last name field. The system doesn't want me to have a number in the last name. Ideally it should let me put whatever I want in my name, but putting that issue to one side for the moment, a simple way to communicate this error is to say: 'You can't have numbers in your name'. Instead the error message reads 'Character 2 at position 6 is not allowed.'

Curiously, the offending number is actually in position 7. However, this error has been written by a developer — and if you're a developer, you start counting from zero, not from one.



The image shows a screenshot of the Yelp website's profile creation interface. At the top right, the Yelp logo is visible with the tagline "Real people. Real reviews.®". Below the logo is a red error banner with a "STOP" icon and the text "Character '2' at position 6 is not allowed.". The main form is titled "Create Your Yelp Profile" and contains two input fields. The "First Name:" field contains the text "Bill". The "Last Name:" field contains the text "Gates 2nd".

An error message that only a programmer could love.

Don't blame the user

Designers would be wise to follow Alexander Pope's aphorism, 'To err is human; to forgive, divine.' The test of a user centred error message is how well it succeeds in helping

the user save face after an error has occurred. A user's instinctive reaction to an error message is, "Aargh! I did something wrong!" Like a polite butler, the system should respond, "Oh no sir... that was my mistake." Nobody wants to feel stupid when using your system. So check that error messages are written in a non-derisory tone, do not blame the user and avoid phrases like 'user error' at all costs.

The example below comes from an earlier version of the Abbey National online banking web site (before the bank became Santander). As with many of the other examples in this article, this particular error message has a number of problems: for example, it doesn't help me resolve the issue (other than providing a helpdesk number) and it uses an error code rather than a precise description of the error. But my reason for including it in this section is because it does such a good job of making me feel stupid. Why is the word 'error' presented in such a large font? (Presumably so the fraudster shoulder surfing my online banking session can realise I'm totally clueless and ripe for the taking).



This error message appeared at an online banking site after a problem signing in. It contains the word 'error' in a massive font implying that the user has made a mistake (when in fact this was a system problem).

Another example with which we're all familiar is the '404-page not found' error that appears when you follow a broken link. Like many of the other examples I'm showing, the following example contains more than one problem — but my reason for highlighting it here is because of the opening phrase: 'you did something wrong'. Hold on: I clicked on a link in your web site and got this page. Don't tell me this is my mistake.

404 Not Found - BeerAdvocate

We're sorry, you did something wrong -OR- the requested URL could not be found.

Some possible reasons for this error are listed below:

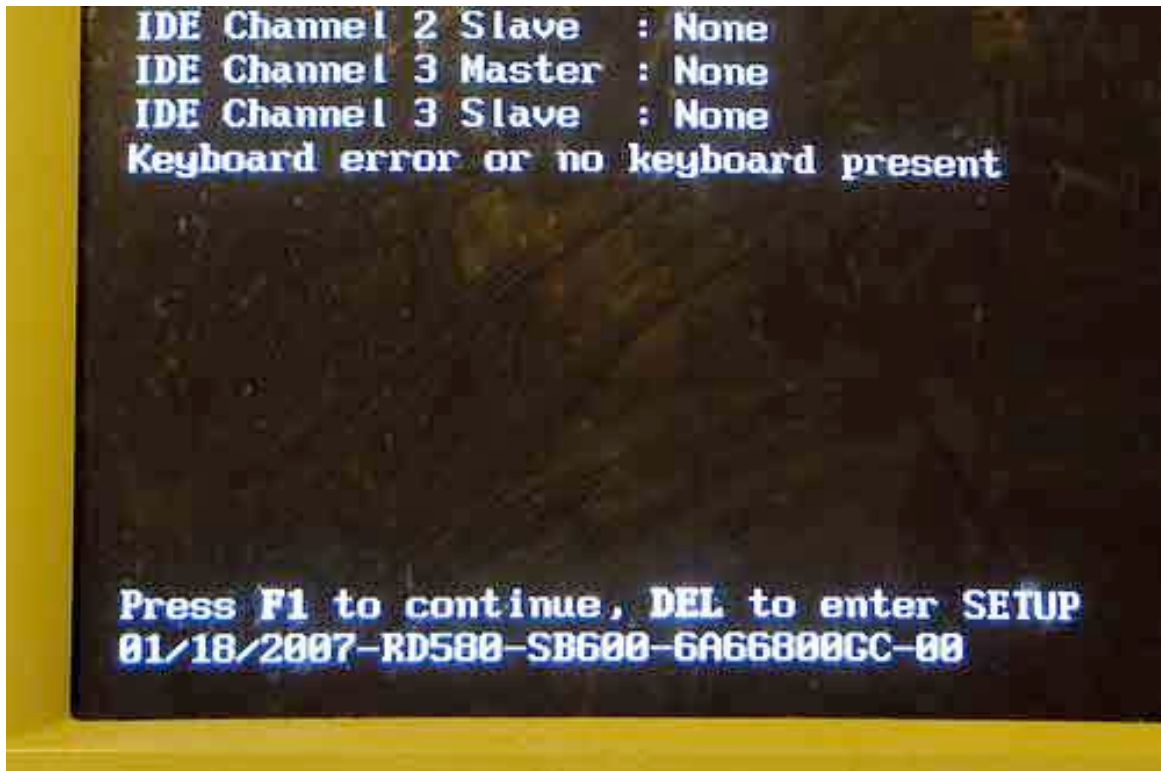
- The page you've requested no longer exists.
- Perhaps you mistyped the URL?

The day I broke the Interwebs.

As well as deflecting blame from the user, a good '404' page should make suggestions on what users should do next to reach the page they are after, include a search box, and perhaps include links to the most popular pages in the site.

Don't be stupid

Stupid error messages are error messages that are self-contradictory or that otherwise contain errors of their own. Perhaps the most famous example is the classic 'Press F1 to continue' when a PC is started without a keyboard:



The error message reads, 'Keyboard error or no keyboard present. Press F1 to continue, DEL to enter SETUP'. Photograph by [Brent Nashville on flickr](#). Some rights reserved.

Similarly, you need to ensure that your error messages are free of spelling errors or typos. An error message with a spelling error is the very pinnacle of idiocy. It's like trying to correct someone's grammar and getting it wrong.

The example below shows a screenshot from a car park payment machine. Note the typing error. (Again, this isn't the only problem with this error message: it seems to imply the user has made the error with its 'card is illegible' statement):



The text on the error message contains a typing error ('stipe' for 'stripe'). Photograph by [Justine Sanderson \(Titne\) on flickr](#). Some rights reserved.

Similarly, the error message below includes HTML formatting that simply makes the error message very hard to read.



This pop-up message from the lastminute.com web site leaks html formatting commands into the error message itself.

Wrapping up

To review the 6 principles again:

- Be visible
- Be precise
- Give constructive help
- Speak the user's language
- Don't blame the user
- Don't be stupid

Nobody likes to see an error message. But follow these principles and you'll remove some of the misery from the experience.

4 ways to prototype faster

Ritch Macefield

“Lean UX” is the new black. We can summarise the philosophy behind it by saying: If a picture is worth a 1000 words, then a prototype is worth a 1000 pictures (with apologies to Ben Shneiderman). But given that we are increasingly working in environments where we need to deliver more with less, how can we speed up the process of prototyping?

We can prototype faster by following 4 key principles:

- Start prototyping with paper
- Use one electronic tool, not several
- Use a prototype that generates specifications
- Support collaboration

I'd like to describe how we can apply these principles in a typical Agile environment.

Start prototyping with paper

The best way to speed up your prototyping efforts is to start with paper. One of the tenets of Agile is that working software is a key measure of progress on a project. Sadly, developers sometime interpret this to mean that they should be creating computer-based prototypes right from the start.

Here's the best kept secret in prototyping: the biggest mistake you can make is to start prototyping in front of a screen.

The problem with screen-based prototyping is that it does not encourage you to fully explore the design space. Instead, you tend to latch onto 1-2 design ideas and prototype those.

In the early phase of design — whether you're designing a new system or a new workflow — you don't want 1-2 ideas: you want dozens. That makes paper and Sharpies the prototyper's new best friends. In this initial phase of design, you should create dozens of paper interfaces, iterate on the ones that have value, and then develop interactive paper prototypes that you can use to test out the design with users.

Use one electronic tool, not several

When you move to electronic prototyping, there's a second way to improve your efficiency: use the same electronic tool throughout. I often see design teams start their prototyping effort with tools like Keynote and Balsamiq, move to Photoshop or Fireworks to create prototypes with higher visual fidelity, then turn to Dreamweaver or Flex to add interactivity for final usability testing.

One problem with this approach is that designers need to recreate the design in their favoured tool, which creates inefficiencies and slows down development. It also creates problems in managing all the software and files: for example, using tools like Visio to define user journeys which relate to screens designed in Photoshop which have been annotated in MS word.

You'll be much more efficient if you standardise your prototyping efforts around a single platform.

Use a prototype that generates specifications

A third way of making the entire process more streamlined is to ensure your prototype contains the detailed specifications that developers need to implement the design. The devil, as they say, is in the details.

A common mistake made by novice prototypers is to hand their prototype to the developer and ask them to code it up. Those without real world experience of Agile may be tempted to think that Agile eliminates the need for documentation — in fact, it only reduces the amount of documentation. Agile may not need massive requirements and specification documents, but it still needs some information to supplement prototypes so that we can define the goals for an Agile sprint (e.g. hex values for colours, and the contents of drop down boxes).

You could cut and paste Photoshop images into Powerpoint, Word or Visio for annotation. But this is very laborious and you'll need to update the specification document with any changes to the prototype's design. (Indeed, this is such a nightmare that I have known major corporations shy away from making sensible changes to prototypes purely because updating the specification documentation would have been so time consuming!) Instead, choose a prototyping tool that allows you to generate these specifications from the prototype or that allows developers to easily inspect elements so they can see what they need to implement.

Support collaboration

You can make the whole process even more efficient by using prototyping tools that support collaboration.

The ideal prototyping tool:

- Supports multi-user prototyping. Your tool should support file management, revision control, access rights and data integrity (ensuring two or more people are not updating the same work at the same time).
- Allows you to easily distribute the prototype for feedback. Some organisations are still distributing prototypes by simply emailing a zip file containing lots of individual files. This means feedback takes the form of un-coordinated email replies. Instead, choose an electronic tool that helps you structure this feedback.

My recommendation

Those of you that know me will also know that I'm a great fan of Axure and iRise, both of which address many of the issues I've listed above. These tools have been specifically engineered to meet the demand of prototyping in a modern software development environment.

If you're in an environment where you're having to do more with less, and do it more quickly, then try adapting this Lean UX idea to your own practice.

Why you need a user experience vision (and how to create and publicise it)

David Travis

Many design teams launch into development without a shared vision of the user experience. Without this shared vision, the team lacks direction, challenge and focus. This article describes how to use the 'Design the Box' activity to develop a user experience vision, and then describes three ways of publicising the vision: telling a short story; drawing a cartoon showing the experience; and creating a video to illustrate the future.

What is a user experience vision?

Look at the desk of a typical user experience designer and you'll see the obvious paraphernalia of the job. Personas. Wireframes. Usability test reports. One thing you're less likely to see is evidence of a user experience vision: an idealised view of the experience that users will have with the product, set a few years in the future. This vision captures the critical elements of the user experience and articulates the "winning idea" — focusing on the experience and downplaying the technology required to get there.

What happens when you don't have a user experience vision?

Three problems occur with design teams that don't have a vision of how their product will be used in the future.

1. The team lacks direction. This causes problems when the budget is cut or the product needs to ship earlier than planned. The team needs to make design trade-offs, but there is no consensus on what can be cut and what must stay. Inevitably, what gets cut is the feature that's hardest to implement — when this may be the very feature that's needed to make the product a success with users.
2. The team lacks challenge. A product without a user experience vision has little relevance to real people's lives — and so the design team don't feel inspired to do great design. Steve Jobs reportedly recruited John Sculley from Pepsi-Cola by asking, "Do you want to sell sugar water for the rest of your life or come with me and change the world?" Without an inspirational challenge, you'll find people on the design team fail to get excited about the product.
3. The team lacks focus. Because there is no shared vision of what really matters in the product, each person on the team creates their own priorities. This leads to disagreements in design meetings and risks derailing the design effort.

How to create a user experience vision — “design the box”

It's not easy to start creating the vision for a product when you're many months into designing it. But here's one activity (described on Joel Spolsky's blog) you could try now to get this on your design team's agenda. This is a simple and fun technique to involve the whole design team in the development of the vision and ensure you get consensus.

In this activity, you imagine that your product will be sold on a shelf in a retail store. Your job is to design the packaging for the product. (It doesn't matter if your product is 'virtual', like a web site, the process still works fine). Working in small teams, you “design the box” for your product. Here are some hints you can give to the teams:

- Agree on the most important message — the key takeaway — that the box should convey.
- Invent a name for the product that captures the winning idea.
- Include a picture of the product being used (draw a sketch or take a photo).
- List the main features of the product.
- List a handful of benefits that users will get from the product.
- List the requirements for the operating system (or the operating environment).

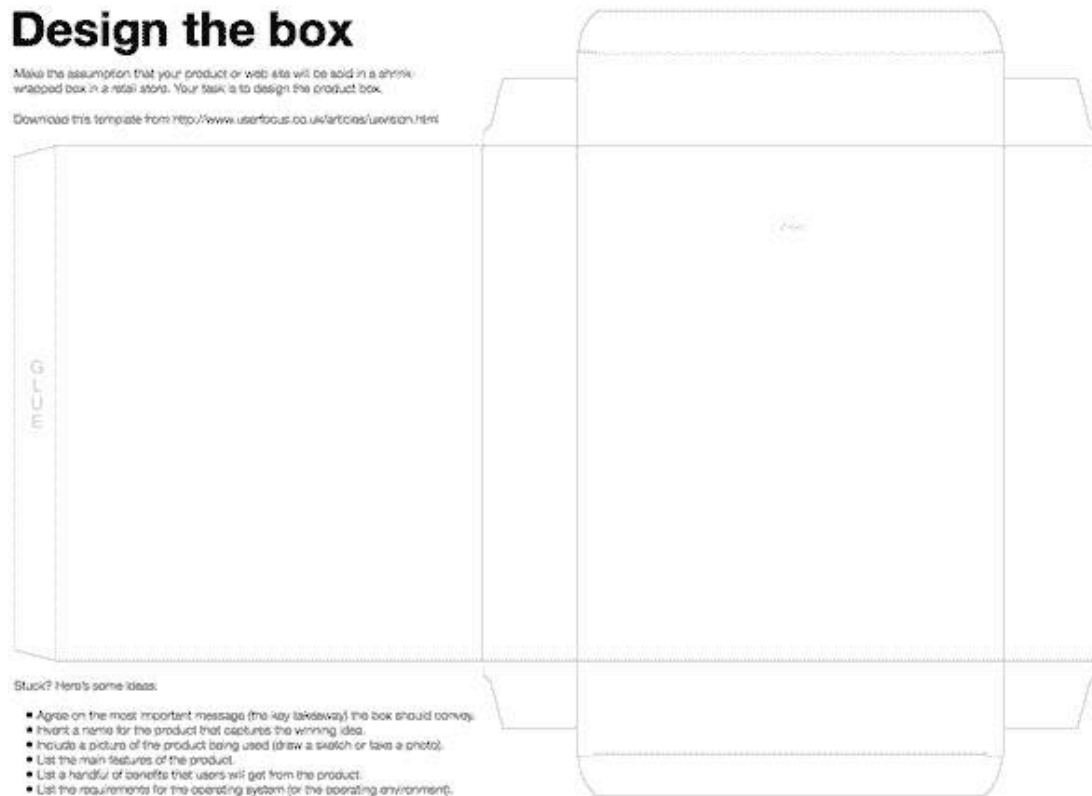


A presentation after completing the “design the box” activity. Note that the emphasis is on presenting the vision, not on the quality of the box’s artwork. By [4david on flickr](#) (some rights reserved).

To give people a head start, you need to supply some box templates. You can either download and print our template for the “design the box” activity (for best results, print the template on card stock), or you could recycle empty cereal packets (carefully break the seams and then turn the box inside out).

After 30 minutes or so, each team presents their box to the larger group. Restrict each presentation to 2 minutes — an elevator pitch — to ensure that each team focuses on the essence and doesn’t get lost in the details. For an even stiffer challenge, get teams to summarise their key message in a tweet (140 characters or less).

If your experience is typical, you'll find some areas of agreement and many areas of disagreement. That's fine. The purpose of the activity is to discuss the findings and reach a consensus. You'll find that the user experience vision emerges in the subsequent discussion.



A template for the "design the box" activity.

If you work in a small user experience team, you might find there aren't enough voices for a good 'design the box' activity. In that case, you could try 'writing the press release' instead. Ian McAllister (who uses this approach at Amazon) describes the approach like this:

For new initiatives a product manager typically starts by writing an internal press release announcing the finished product. The target audience for the press release is the new / updated product's customers, which can be retail customers or internal users of a tool

or technology. Internal press releases are centered around the customer problem, how current solutions (internal or external) fail, and how the new product will blow away existing solutions.

The idea is that the press release is written before any development work starts, a technique that Amazon call 'working backwards'.

3 ways to publicise your user experience vision

The "design the box" activity provides the launch pad for your product's user experience vision. Now you need to refine the vision and present it so that management, the development team and new recruits understand it. Here are three ideas:

- Tell a story
- Draw a picture
- Create a video

Tell a story

The simplest approach to articulating the user experience vision is to tell a story. Stories are powerful ways to communicate design concepts because they focus on the user's experience with your product and gloss over the implementation details. This means they make the "winning idea" self evident. Here's an example of a user experience vision for a home media centre:

The Norris family are big movie fans but they are frustrated by having to set up and switch between the various devices when they want to watch a movie: the PlayStation, the digital video recorder, the TV set and the three computers. So the excitement in the living

room is palpable as they unbox their new media player and connect it to their 42" plasma TV set. The new media player will effortlessly find all the movies that reside on all the devices in their household and provide a single interface for playing them back.

A story like this captures the winning idea in a few words and makes it clear where the design priorities lie.

If you're serious about storytelling, a great resource is Whitney Quesenbury and Kevin Brook's, *Storytelling for User Experience: Crafting Stories for Better Design*.

Draw a picture

Although written stories often work well as a communication tool, you may feel that you need more visual ways of communicating the user experience vision.

One simple way to achieve this is to create a cartoon showing the experience. Cartoons are very accessible, easy to read and easy to share. The example below shows a user experience vision in this format.



A cartoon showing a user experience vision for a mobile application.

If you're worried about your drawing skills, don't be. Simply take photos, and then 'cartoonify' them in an application like Photoshop (applying the 'Cutout' filter first, then the 'Poster Edges' filter, will give you a realistic cartoon effect). Alternatively, import your photos into an application like Comic Life which has its own built-in filters for turning photos into comics.

If you're unable to take photos, you can use comic clip art resources. One of our favourites is Martin Hardee's Design Comics. This contains free comic characters and scenes

showing people using computers that you can use to develop your user experience vision.

Create a video

This way of presenting your user experience vision requires the most work, but you'll find it's the most compelling. A well produced video can really demonstrate the future today because you can actually show people interacting with your envisioned product.

An excellent example of the genre is Adaptive Path's "Aurora" Concept Video. Another fine example is Apple's 'Knowledge Navigator' video. Jared Spool has written a great piece deconstructing the Knowledge Navigator video, and you can use his insights as guidance on what you should include in your own video.

It's time to step up

One refrain I sometimes hear from people in UX teams is that they are not sure who should take responsibility for building the user experience vision. Isn't this something that Marketing should do, or Senior Management?

Part of the problem is that UX teams often have little ownership of anything at a strategic level. Well, this is your chance to grab a leadership role. Developing a user experience vision will demonstrate to your management that you're not just a tactical expert. It will show them that you can see the big picture too.

Card Games for Information Architects

David Travis

This article reviews 6 simple but powerful research techniques you can use to improve the information architecture of your product or web site. None of these activities requires a computer. You simply need a bunch of cards, a participant and a desk.

Over the last few years we've seen a quiet revolution in user experience research. Participants no longer need to come to a usability lab. Nowadays, we can carry out many user research activities over the web. Although we welcome this change (and have even developed our own remote usability testing tool), user experience research is fundamentally straightforward. There's a lot you can do with the simplest of tools.

Exploratory card sorting

Many people are familiar with card sorting as a method of finding out how your users classify the information in your domain. You need this information to create the menu system for your application or the information architecture for your web site. Figure 1 shows a card sort in progress.



Figure 1: A participant carries out an open card sort.

How to carry out exploratory card sorting

- Prepare a stack of cards. Each card contains a title (usually the title of a function or a page within your web site). In my card sorts, I also include a short description of what that function is used for (this is optional but helps avoid simple keyword matching). Card sorting works well when you have more than 30 and less than 100 cards. See Figure 2 for an example of a printed card.
- Ask around 15 users to sort the cards into groups that make sense to them.
- Once the cards have been sorted into groups, give each user a stack of blank cards. For each group, users take a blank card and write on the card a description of what

makes that group a group (these will become the proto-headings of your navigation structure).

- Analyse the data to (a) find out the cards that users place together, (b) the terms people use to describe the groups of cards, (c) cards that appear in multiple categories and (d) how the results compare with the existing (or proposed) design. Software exists to make the analysis easier — I'd recommend SynCaps, version 1 of which is now a free download.

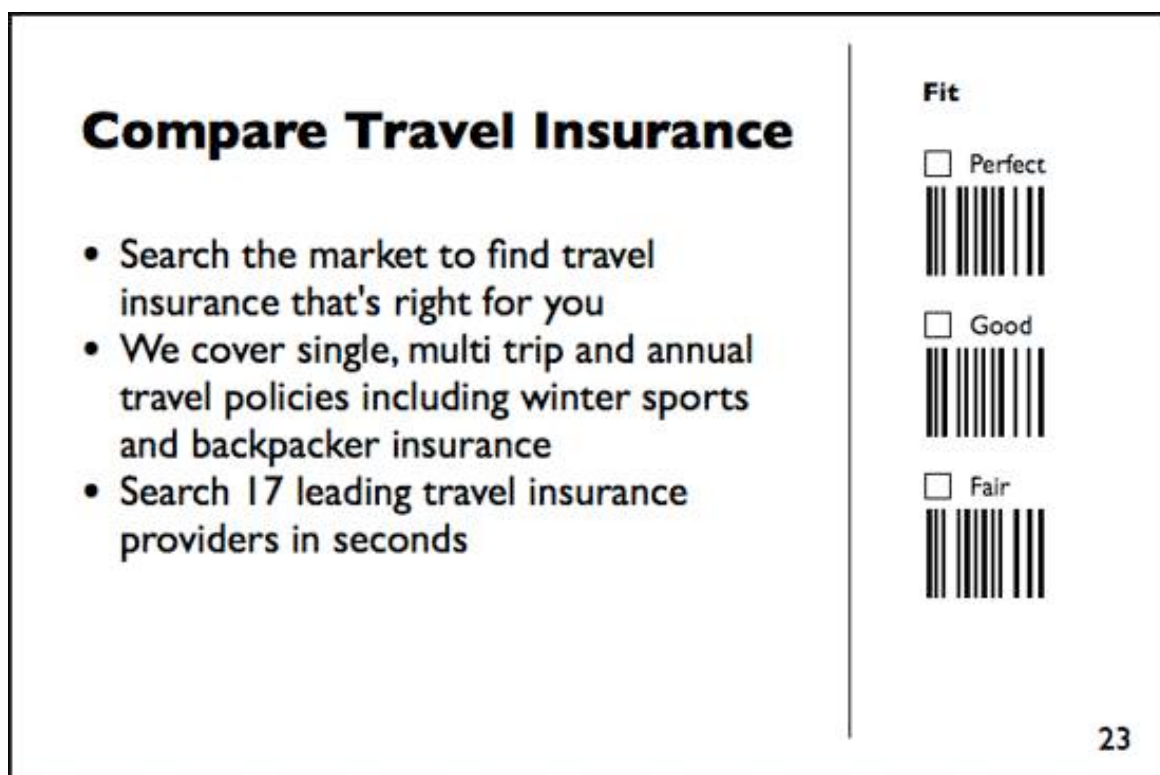


Figure 2: This card is barcoded to make data entry easier and has a goodness-of-fit checkbox for users to rate how well the card fits in the group. (Printed using the CAPS system).

The process I've just described is called an "open" card sort (for much more detail, I'd recommend Donna Spencer's book on card sorting). You can also carry out a closed card sort where you give people the group headings and ask them to put

the individual cards in each group. This tests the quality of the taxonomy you have created.

But if you're interested in where in the navigation structure people would go to complete a task, you should use a tree test.

Tree test

Use a tree test to check if people can actually find stuff in your navigation structure.

How to carry out a tree test

- Prepare a stack of around 10-20 cards. Each card contains a short description of a likely goal that your web site supports. For example, "Find an exercise bike" might be common task for a web site that sells fitness products. Tasks where people are asked to find stuff work well with a tree sort.
- Prepare a set of group headings — these will be the headings from your existing navigation structure or derived from an open card sort. Then, on the back of each card, write down the sub-groups that appear in that group. For example, on one side of the card, you might have the group heading, "Sports & Leisure". On the reverse of the card, you might have the sub-groups, "Fitness", "Camping & Hiking", "Cycling," etc.
- Lay out the group headings in front of the user in a grid. Give users the task cards. Ask them to pick the group card they would use to start that task. Then turn over the group card and ask them to pick the sub-group they would use to complete that task.

- Analyse the data to find out the percentage of users that choose the correct option for each task.

Trigger word elicitation

Use trigger word elicitation to identify the words or phrases that will encourage users to click on links. You need to identify trigger words because users arrive at a page on your web site, hunt around for a link that seems the best fit to their goal and then they click on it — and if your site doesn't contain the trigger word, they hit the back button.

How to carry out a trigger word analysis

- Prepare a stack of around 10-20 task cards, as with the Tree Test. Each card contains a short description of a likely goal that your web site supports. For example, "Book a romantic meal for two" might be a common task for a restaurant web site close to Valentine's Day.
- Ask users the words or phrases they would expect to find on a web site that supported that task. For example, you might get phrases like, "Book a table", "Make a reservation" and "Put your name down".
- Analyse the data to identify the common words used for each task. Make sure your web site contains those words to direct users towards the appropriate content.

Web board

Use a web board to find out where users expect to find your functions. This works well for a web page where navigation items appear in headers, footers and page margins as well as the more conventional menus. You need these data because if

your functions appear where people expect, they will find them more quickly. The Software Usability Research Laboratory have published a case study using a similar technique.

How to use a web board

- Take a screen shot of a blank browser page. Open the image in your favourite drawing application and overlay a 5 x 5 grid on the page (Figure 3 shows an example). You want the print out to be as large as possible, so print it on A3 paper or use a photocopier to scale up an A4 or Letter-sized page to A3.
- Prepare a stack of cards. Each card contains a title of the function or link and a short description of what that function is used for (as with an card sort).
- Tell users that the browser page represents a page from your web site. Ask users to place each card on the grid in the approximate location where they would expect to find it. For example, you may find that users place a 'Back to home' card in the top left square, and a 'Sign into your account' card in the top right square
- Using the letters and numbers on the grid, note down where users expect each function to appear. Analyse the data to identify areas of agreement across participants. Make sure your functions and links are placed in these locations.

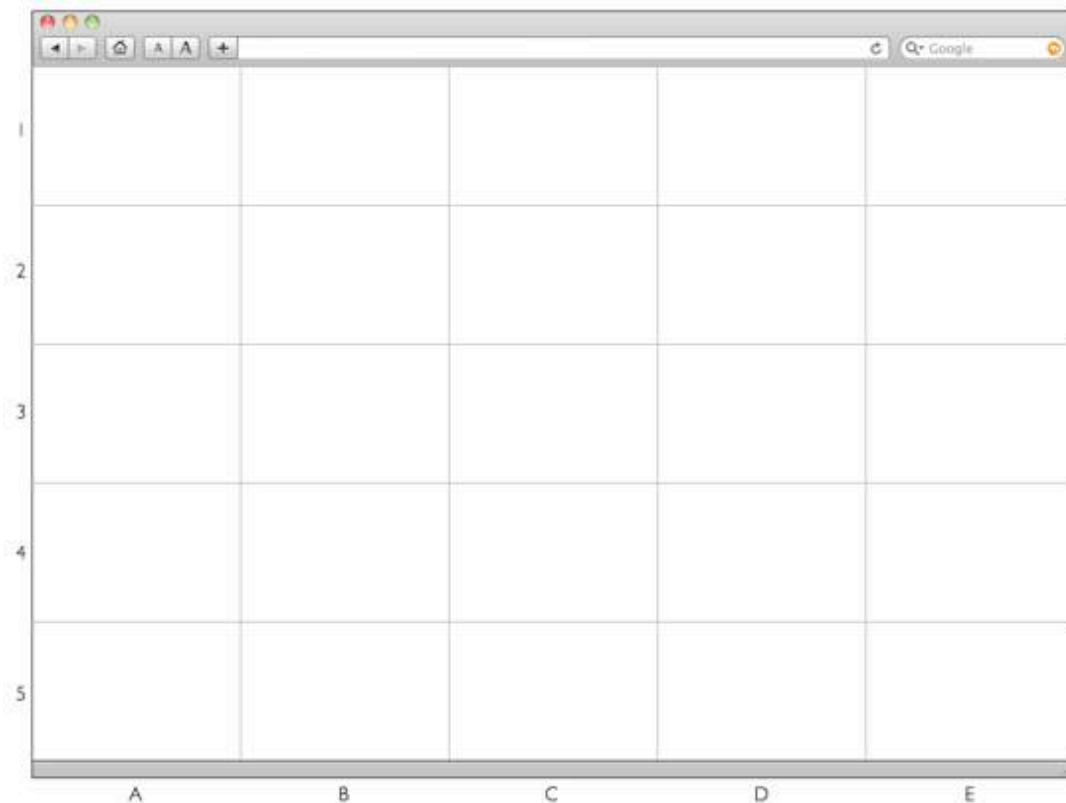


Figure 3: Browser grid used for a web board.

Function familiarity test

Use a function familiarity test to gauge how often people use functions within your application. You need this information to discriminate frequently used functions from infrequently used functions.

How to carry out a function familiarity test

- Prepare a stack of cards. Each card contains a title of the function and a short description of what that function is used for. For example, for a mobile phone you might have “Address book”, “Ring tones”, “Network”, etc.
- Ask users to sort each card into three piles: functions I use frequently; functions I use sometimes; and functions I rarely or never use. (This is easy to adapt: for example,

you could change the test to measure levels of understanding by naming the piles, “Functions I know how to use”, “Functions I can use by muddling through” and “Functions I don’t understand”).

- If a function appears in the “use frequently” pile, give it 5 points. If it appears in the “use sometimes” pile, give it 2 points. After you’ve tested all participants, add up the scores to identify the most used functions in your interface.

Swap Sort

Use a swap sort to find out the most important functions, features or tasks within your interface. You need this information to know how to prioritise content.

How to carry out a swap sort

- Prepare a stack of cards. Each card contains a title of the function, feature or task and a short description of it.
- Ask users to read through the cards and fish out the 10 most important cards for them. Put the other cards to one side — you won’t be using them anymore.
- Ask users to place the 10 “most important” cards vertically in front of them. Ask users to rank order the cards by swapping adjacent cards, putting the most important ones higher up in the list, until no more can be swapped.
- Give the card placed first in the list 10 points, the card placed second in the list 9 points and so on — so the card placed last in the list gets 1 point. After you’ve tested all your users, total the points for each card and

use the results to identify the most important functions in your interface.

Tips for writing user manuals

Philip Hodgson

User manuals have a bad reputation. In a recent USA Today poll that asked readers “Which technological things have the ability to confuse you?” user manuals came out top! Increasingly companies are rethinking the way they approach user manuals. Here are some suggestions for improving the usability of user manuals based on our experience writing them.

These guidelines are based on:

- Best practice principles.
- Principles of good information design.
- Aspects of human perception, cognition and psychology as it pertains to reading.
- Our own experience of user testing various kinds of user manuals and documentation and seeing what works and what doesn't.

We have arranged the tips into the following sections:

- General guidelines
- How to create a great first impression
- How to enhance findability
- How to give instructions
- How to design individual pages

- How to design the physical manual

General guidelines for user manuals

- Provide a real (physical) user manual with the product: don't make people read a pdf.
- Make sure the instructions actually map on to the product in all respects.
- Include a one-page quick start guide.
- Present instructions as step-by-step procedures.
- Tell the user what functions there are, and what they are for — not just how to use them...
- ...but avoid marketing waffle (they already bought the product!)
- Ensure that the writers are part of the product design team.
- Write the user manual in synch with the product's development timeline — not under pressure of shipping deadlines.
- Make sure the writers have the product, understand the product, and actually use the product as they write.
- Consider the needs of disabled users (i.e., low vision, colour-blind) and provide alternative manuals in Braille, large print, audio etc.
- User-test the product and the user manual with real users (including disabled users).

How to create a great first impression

Many users never actually get as far as the user manual. It is often tossed aside as being either secondary, or just too difficult to deal with. When this happens, the user, the product and the writing team all suffer in some way. In order to get past this point the user manual must make a strong and positive first impression. These guidelines can help.

- Avoid a text-book look (landscape formatting can be less threatening).
- Use paper that is commensurate with the quality of the product.
- Make purposeful and effective use of colour.
- The user manual should not be too big or too heavy...
- ...or too small or too flimsy.
- Make effective use of pictures and diagrams.
- Provide lots of white space.
- Use a clean, readable san-serif font.
- Include a help-line number.
- Use a single language.

How to enhance findability

Users quickly get frustrated when they cannot find what they are looking for in the user manual. Often this is due to the fact that the key words the writer has used are not the key words that users may search for. Here are some guidelines that will help users find what they are looking for.

- Organize information hierarchically.
- Code the hierarchy with tabs, colours etc.

- Divide into sections ordered by:
- Chronology of use.
- Frequency of use.
- Functional categories.
- Expertise level (beginner vs. expert user).
- Denote importance by using contrast, colour, shading, boldening etc.
- Work with real users to identify likely key words (these can be learned during usability testing).
- Provide a key word index using the terminology of the user.
- Ensure that the index includes likely synonyms.
- Provide a glossary of technical terms.
- Include a (genuinely useful) trouble-shooting section.
- Use colour-coding to aid navigation.
- Make the quick start guide readily accessible.
- Avoid unnecessarily cross-referencing to other parts of the user manual.
- Avoid duplicate page numbering in multi-language guides (better still, avoid multi-language).
- Clearly display the help-line number.

How to give instructions

Clearly this is the primary role of the user manual. It is critical that the instructions are easy to read and are understandable by all users. Many user manuals have instructions that are incomplete, incorrect, or simply have no

bearing on the actual product. Here are some guidelines to help make instructions easy on the user.

- Provide step-by-step sequences in the correct order.
- Follow the timing and sequencing of the actual operations .
- Provide visual stepping stones (e.g. Step 1, Step 2 etc.)
- Avoid lengthy paragraphs.
- Use everyday words and terms: avoid jargon.
- Explain what a function or feature is for (in basic practical terms) as well as “How to” instructions.
- Check that the instructions match the actual product.
- Explain symbols, icons and codes early.
- Avoid creating dead-ends.
- Avoid patronising the user.
- Do not assume the user has prior experience or product knowledge.
- Usability test the instructions alongside the product using naive users (not designers or product experts).
- Write in the present tense and the active voice.
- Write the steps to task completion while doing the actual task on a real product. Have an independent user then follow the steps (literally) with the product and check that:
 - It is easy to work through the task from start to finish.
 - It is easy to break out of task and get back in.
 - It is easy to jump into the user manual half way through a task.

How to design individual pages in the user manual

In addition to effective instructing, the use of colour, the text and fonts used, and the icons and graphics can all either make for an easy experience or can derail the user. Here are some suggestions.

- Ensure that font size is adequate (use at least 12 point font).
- Ensure high text-to-background contrast (black on white is best).
- Use san-serif fonts.
- Avoid using multiple font styles.
- Font weight can be used sparingly to denote importance.
- Use colour coding consistently.
- Provide plenty of white space between sections and around images and paragraphs.
- Provide a section (or margins) for the users to make their own notes.
- Use consistent layout from page to page.
- Test your use of colours to ensure they can be read by colour-blind users.
- Avoid using saturated blue for text and small detail, and never use blue on a red background.

How to design the physical manual

User manuals are used in many different kinds of environments: they may be used indoors or outdoors, they may

be used with good light or with dim light, they may be used in a comfortable and user friendly setting or in an environment that is hostile or even dangerous. Here are some basic guidelines to ensure your user manual will survive actual use.

- Ensure that the user manual can lie flat on a work surface when opened.
- Consider the environment of use and if necessary provide a robust user manual.
- Consider whether the user needs to hold the user manual and work at the same time.
- Provide durable covers and pages.
- Consider whether the user manual needs to resist water, oil, dirt, grease etc.

Five kinds of 'alt' text

David Travis

There are five different classes of image used on web pages and each class of image requires a different approach to writing the 'alt' attribute.

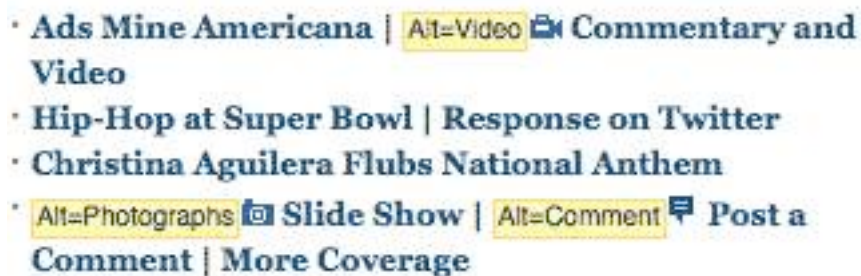
Virtually every web designer I speak with is familiar with the 'alt' attribute: the part of the html 'img' tag that you use to provide an equivalent alternative for people who are unable to see the image. This includes people who are using a screen reader or people who are browsing the web with images turned off. What's less commonly known is that there are five different classes of image used on web pages and each of those images requires a different approach to writing the 'alt' attribute.

The five different classes are:

- Eye candy.
- Clip art and stock images.
- Images that express a concept.
- Functional images.
- Graphs, complex diagrams and screenshots.
- The 'alt' text you write will be different for each of these classes of image.

Eye candy: use an empty 'alt' attribute

Most web pages are full of eye candy, like horizontal rules, glyphs and spacer images. The purpose of these kinds of image is to make a web page look more attractive but if you describe these images in the 'alt' attribute it just creates auditory clutter for people who are using a screen reader. People using a screen reader don't need to hear, 'spacer image', 'glyph image' or 'small green bullet' when they are browsing a web page since this contains no useful information.



In this screenshot from the New York Times, I've used a feature in the Firefox Web Developer toolbar to overlay the 'alt' text for each image. This shows that the designers have chosen to write an 'alt' attribute for the video camera, still camera and comment glyphs. These images are eye candy and do not need any alternative text. A blank 'alt' attribute would be the correct choice here.

For these types of image, you should use an empty 'alt' attribute, that is, alt="". The reason to use an empty 'alt' attribute is that this is a signal to the screen reader to skip over the image. When a screen reader comes across an image with an empty 'alt' attribute, it ignores the image: it's as if the image was never there in the first place.

Why not omit the 'alt' attribute entirely? You can't do this because the 'alt' attribute is a required part of the 'img' tag. If you omit the 'alt' attribute, your page won't be valid HTML. This is like writing a sentence and leaving out the full-stop at

the end: it's ungrammatical. As a consequence, screen readers assume that images with a missing 'alt' attribute were created by the clueless and so make a valiant attempt at describing the image. The usual approach that screen readers take is to read out the name of the image (such as 'logo.gif') in the forlorn hope that the developer at least used some meaningful text in the filename.

Clip-art and stock images: use an empty 'alt' attribute

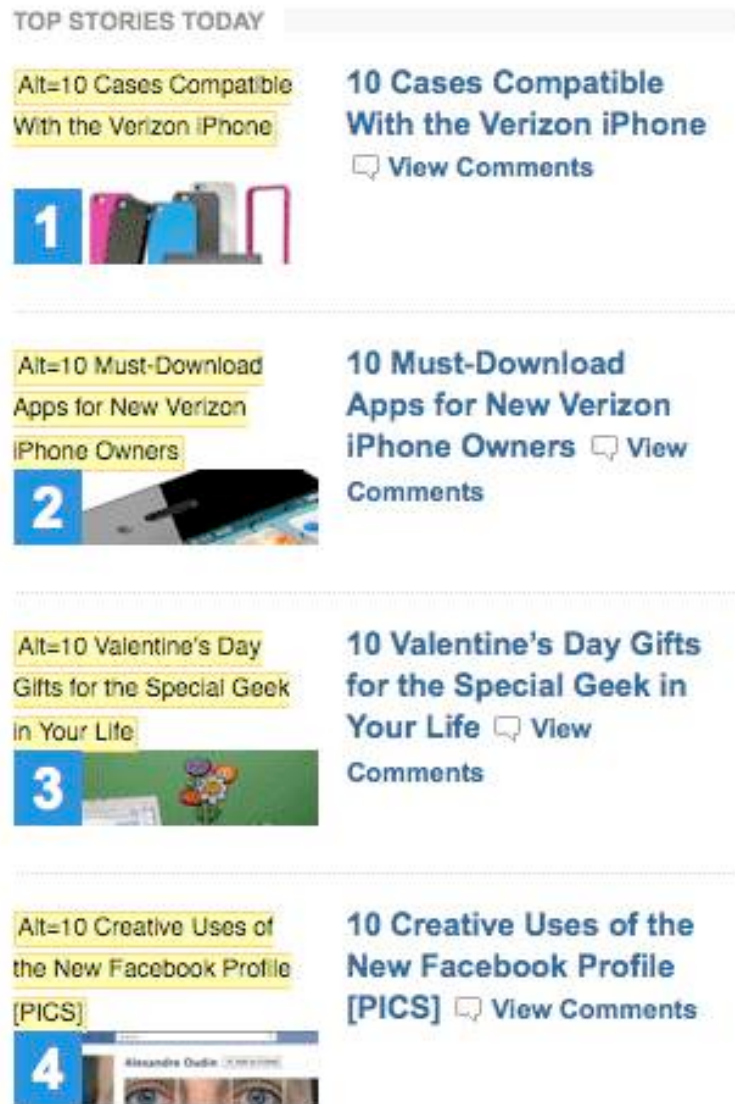
Clip-art and stock images are often used to illustrate an article or news story: you might find a thumbnail version of the image used on the home page and a larger version of the image used in the article itself.

For the vast majority of these images, a blank 'alt' attribute is sufficient. This is because these images are nearly always next to text that is equivalent to the image. The image below shows a good example from the Apple Store. You'll see that the designers have correctly used an empty 'alt' attribute for the images, since to add 'alt' text would simply clutter the page when listened to by a screen reader.



This screenshot from the Apple Store shows a series of stock images of Apple products, like the iPad, the iPhone and the iPod shuffle. Again, I've overlaid the 'alt' text and this shows that the designers have correctly used a blank 'alt' attribute for these images. This is correct, because each image is next to a text description that is equivalent (such as 'iPad from £439').

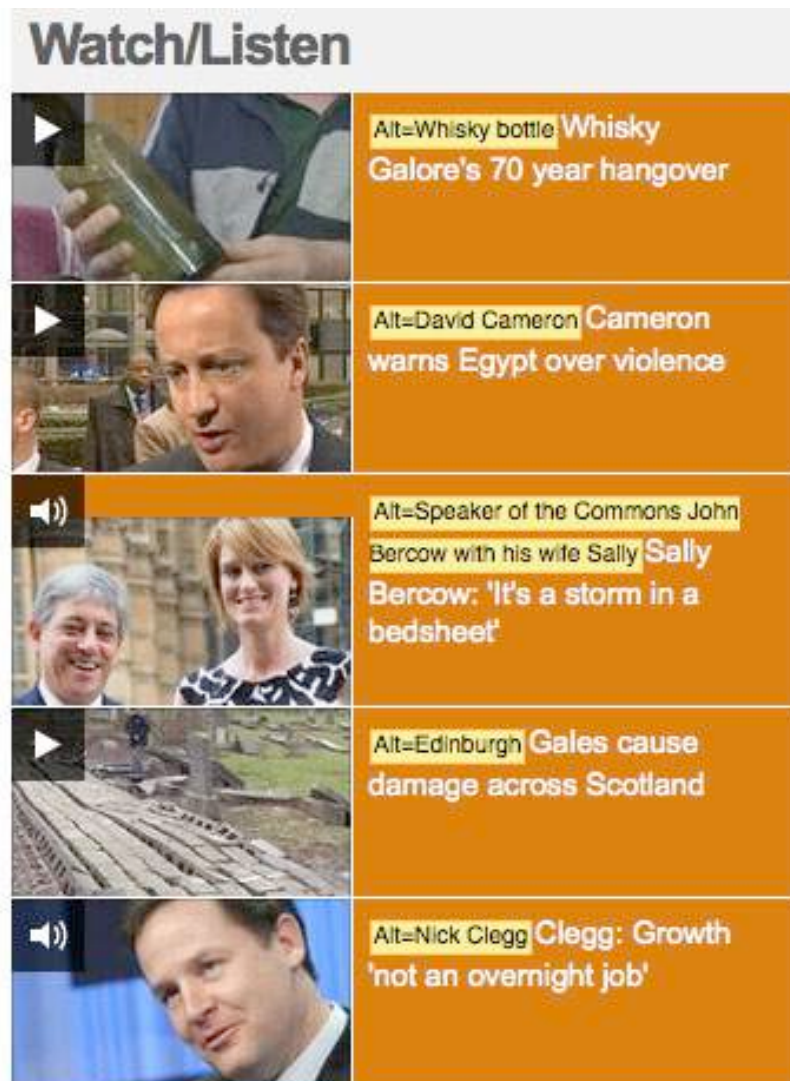
Compare this approach with that used on the social media news blog, Mashable. Here you can see that the 'alt' text is a duplicate of the text that's next to the image. If you're a screen reader user, this creates auditory clutter: you will have to listen to the same text twice. The correct approach here would be to use a blank 'alt' attribute.



This screenshot from Mashable shows a series of thumbnail images that have been used to illustrate a list of top stories. Again, the 'alt' text has been placed next to each image. This shows that the designers have simply repeated the adjacent text in the 'alt' text: for example, there is an article titled '10 Creative Uses of the New Facebook Profile [PICS]' and the 'alt' text simply repeats the text. All of these 'alt' attributes should be blank.

Many news sites fall into this trap. The following example comes from the BBC's site. The images in this screenshot add little, if anything, to the text that's next to them and so including 'alt' text simply clutters the audio stream. This example also shows another common blooper when writing

'alt' text: sometimes designers feel the obligation to convey more information in the 'alt' text than exists in the image alone. For example, the 'alt' text on one of the images reads 'Edinburgh', but few (if any) sighted users would realise that this is a picture of Edinburgh. 'alt' text should never say more than the picture.



This screenshot from the BBC's web site shows a series of thumbnail images that link to video and audio recordings. The 'alt' text has been overlaid next to each image. Most of the 'alt' text simply repeats what is already said in the adjacent text, and is redundant. One picture has been used to illustrate a story titled 'Gales cause damage across Scotland'. The image shows a fallen wall but the

'alt' text reads 'Edinburgh' which conveys more information than either the image or the adjacent text. All of these 'alt' attributes should be blank.

Images that express a concept: use brief 'alt' text and a caption

These kinds of images are usually photographs of people, events or situations. These images communicate a concept and the job of the web designer is to try to express that concept concisely. However, you need to exercise judgement. When I speak with screen reader users, they tell me that they're not really interested in a web page's mood music. They just want to get to the content. So you need to be really sure that this image is conveying something important and it's not just a stock image.

One way you can make that judgement is to ask yourself: if I couldn't use the image here, what would I write instead? If the answer is, 'nothing', then it falls into the same class as stock images, and you should use a blank 'alt' attribute.

If the answer is something else, then you need to find the right form of words. For example, consider this photograph of James Blake Miller, the 'Marlboro Marine'. Using alternative text that read 'Soldier in Iraq' would fail to convey the power of this image.



This powerful photograph by Luis Sinco is difficult to summarise within the brevity constraints of 'alt' text. In circumstances like these, it's probably best to use concise 'alt' text (such as 'The Marlboro Marine') to flag up to the screen reader user that this is an important image, and then include a caption next to the image. When this image appears on news sites, the caption usually reads: 'A close-cropped shot of a US Marine in Iraq, his face smeared with blood and dirt, a cigarette dangling from his lips, smoke curling across weary eyes.' © Luis Sinco, <http://www.flickr.com/photos/mediastorm/2037179165/>

Although there is no fixed rule on the maximum length of the 'alt' attribute, some screen readers split the 'alt' text into 125 character chunks and so that serves as a useful guideline. Since 125 characters will almost certainly be too brief to be equivalent, you should consider adding a caption to this kind of image.

Functional images: explain the function

If the image has a function or communicates information, then the 'alt' text should explain the function of the image. Here are three examples.

The first example shows icons — a weather widget — used to communicate status information. The screenshot (from sky.com) shows how the ‘alt’ text has been used appropriately to explain the icon.



This screenshot shows the weather forecast for Greater London (from sky.com).

The forecast includes images that summarise the day’s weather: for example, Monday’s forecast has an image of a cloud and Tuesday’s forecast has an image of a cloud with the Sun peeking out. The ‘alt’ text has been overlaid on the screenshot and it shows the appropriate use of ‘alt’ text: ‘Cloud’ for Monday and ‘Sunny intervals’ for Tuesday.

A second example is when you use an image of a button (such as ‘Next’, ‘Previous’ or ‘Search’). In these instances, the ‘alt’ text should say exactly what’s said on the button. It shouldn’t read ‘Search Button’ because the screen reader will announce it as a button. Sometimes the buttons are icons (such as using a rightward-pointing double chevron to indicate ‘Next’), in which case the alt text should simply be what you would write if it was a text link (in this example, ‘Next’).

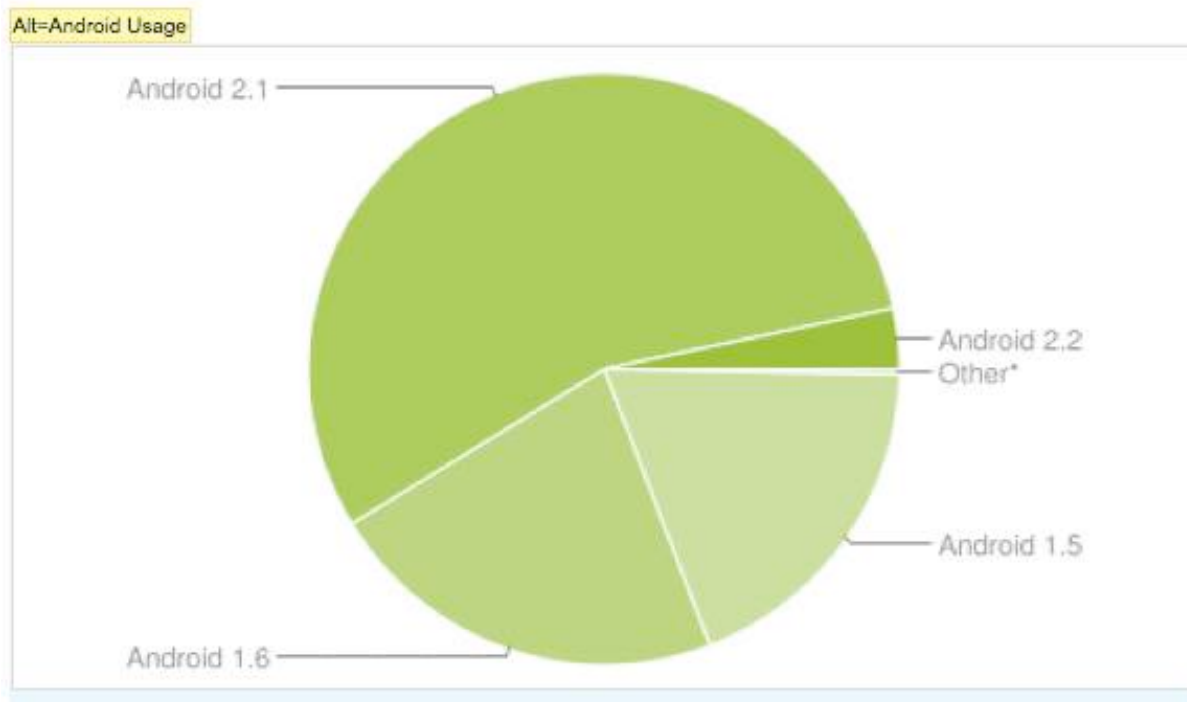
A third example of this kind of image is when a thumbnail photograph is used as a link. Whenever an image is used as a

link it must have 'alt' text, otherwise the screen reader will fall back to announcing the link URL (even if you use an empty 'alt' attribute). Again, the best kind of 'alt' text to use in this example is the same text you would use if you were writing a text link instead of using an image.

Note that these last two examples only apply to images that are standalone links. If you have a link that encompasses the image and some text that follows it (as in the Mashable and BBC examples above), you should use an empty 'alt' attribute. This is because the link is fully described within the text. Adding 'alt' text in these examples just creates clutter. In fact, it's good usability practice to create links that encompass an image as well as the following text, since this increases the size of the whole clickable area and makes them easier targets to acquire (as predicted by Fitts' Law).

Graphs, complex diagrams and screenshots: provide a longer description

Our final class of image is when we show a graph or a complex diagram such as an organisational chart. In these instances, you need to write brief 'alt' text ('Organisational chart for ACME Corporation') and then you must provide a longer, alternative description. This alternative description should be 'equivalent': that is, it should be a standalone description of the screenshot, chart or diagram.



This is a poor example from Business Insider. The problem isn't with the 'alt' text, which is reasonable (although 'Pie chart showing Android usage' might have been better). The problem is that a screen reader user is unable to understand this graph and examine the underlying data. A sighted user can view this graph and quickly identify the most popular version of Android, can see that this version has more than 50% of the market and can see that the next biggest version has less than 25% of the market. A screen reader user cannot get any of this information from the graph.

One solution is to use captions (as I've done for the screenshots in this article) but some sighted users may find this overly wordy. In that case, you could provide a link to another html page where the information is presented in textual form, perhaps as a data table if the image is a graph. In terms of implementation, you should write an appropriately titled link (such as 'Longer description of Android usage chart') and place this immediately after the image. Because you won't want your page of graphs cluttered with all of those 'longer description'

links, you should style them in CSS so that they don't appear in visual browsers.

Summary: The contextual nature of 'alt' text

An important learning point is that 'alt' text is contextual. For example, when there is an equivalent text description close to a thumbnail image, the image should have an empty 'alt' attribute, but not if the thumbnail image is a standalone link. Similarly, it might be fine to have an empty 'alt' attribute when the image is shown as a thumbnail but it may need a caption or longer description when presented as a larger image on the page. As with all design choices, you need to make an informed decision. Avoid the kneejerk reaction of just writing a few words of alternative text and considering your job done.

About the authors

Philip Hodgson



Dr. Philip Hodgson (@bpusability on Twitter) holds a PhD in Experimental Psychology and is a member of the Usability Professionals' Association, the Association for Psychological Science, and the Association for the Advancement of Medical Instrumentation. He has over twenty years of experience as a researcher, consultant, and trainer in product usability, user experience, human factors and experimental psychology.

Ritch Macefield



Dr. Ritch Macefield (@Ax_Stream on Twitter) holds a BA in Creative Design, an MSc in IT / Computing and a PhD in HCI.

He is an acknowledged expert in Axure having led Axure projects for clients like Thomson-Reuters, Dell computers and Vodafone. He was a panel speaker at Axure World 2012, contributed to the book “Axure RP 6 Prototyping Essentials” and founded the Axure RP Pro LinkedIn Group.

David Travis



Dr. David Travis (@userfocus on Twitter) holds a BSc and a PhD in Psychology and he is a Chartered Psychologist. He has worked in the fields of human factors, usability and user experience since 1989 and has published two books on usability. David helps both large firms and start ups connect with their customers and bring business ideas to market.

User Experience: The Ultimate Guide to Usability

Master user experience in this practical, video-based, online training course.

“ The depth and breadth of content covered in this course is seriously impressive. All of the major UX techniques are covered in a way that anyone could take this advice and apply it to their own projects or organisation. If you want to learn how to do user-centred design, this is the course to get.

— *Independent review by Matthew Magain, co-founder, UX Mastery*



“ I already had some in-house, quasi-UX mentoring, but taking this course is what truly opened my eyes to seeing everything in the world from a design and user goals perspective. I promote this course anytime I see or hear anybody asking where they should get started in UX Design/Research.

— *Course review by student Geoff Wilson*

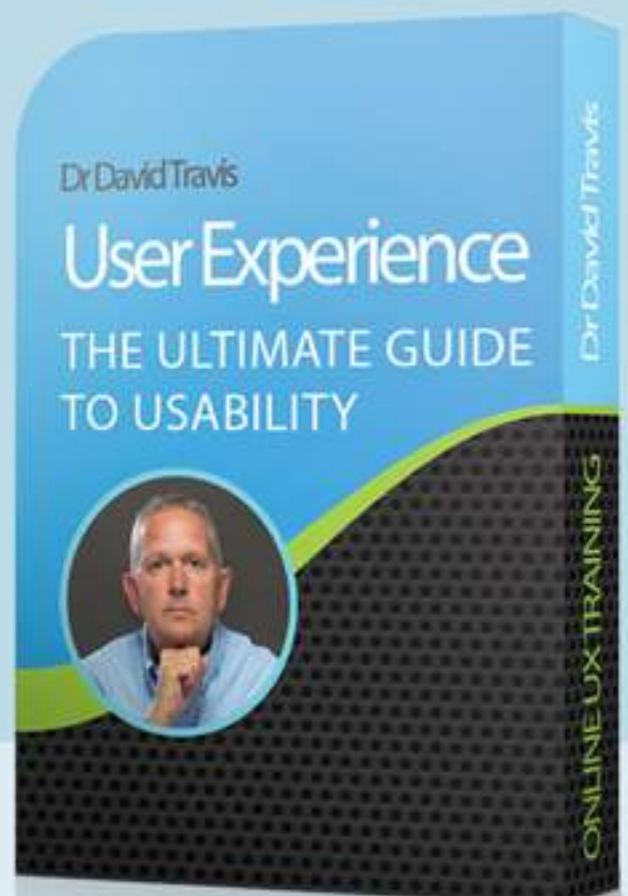
“ Dr. Travis has created a very thorough, very engaging overview of the UX Lifecycle, with lots of great real world examples and war stories from his own considerable experience to illustrate the guidelines and techniques he teaches you. Well worth the time and money.

— *Independent review by veteran usability consultant Dr. Deborah J. Mayhew*

Gain hands-on practice in all the key areas of UX — from interviewing your users through to prototyping and usability testing your designs.

Build a UX portfolio to boost your job prospects as you complete five real-world sample projects.

Gain industry-recognised certification by preparing for the BCS Foundation Certificate in User Experience.



www.uxtraining.net